From Algebraic Abiogenesis to Neural Abiogenesis: Model Compression, Functional Quotients, Recursive Circuit Formation, and the Ground-Up Birth of Mathematical Objects

Douglas C. Youvan

doug@youvan.com www.youvan.ai

July 11, 2026

Mathematics and artificial intelligence are usually described as operating on objects that already exist: numbers, propositions, circuits, geometric forms, concepts, and representations. Algebraic Abiogenesis reverses this order by asking how stable objects can arise from anonymous operations, recursive composition, observational equivalence, and progressive compression. This paper develops that methodology beyond equational algebra into logic, proof systems, NAND circuits, recursive dynamics, complex-plane constructions, fractals, cellular automata, and neural networks. Across these fields, a common generative architecture appears: a primitive substrate is acted upon by anonymous generators; closure under composition or recursion produces a growing ecology of expressions or states; an observation policy identifies behaviorally equivalent outcomes; and stability filters reject degeneracy, finite accidents, redundancy, and nonpersistent structure. The surviving equivalence classes become objects, rules, modules, attractors, or concepts. Neural training and model compression may instantiate the same process internally, converting distinctions among data and activations into functional quotients and reusable computational circuits. The resulting framework suggests that abstraction is not merely represented by intelligent systems but constructed through compression. Mathematical and neural objects may therefore share a common origin in stable equivalence under recursively generated operations.

Keywords: Algebraic Abiogenesis, Neural Abiogenesis, mathematical It from Bit, model compression, functional quotients, representation learning, abstract algebra, logical consequence, proof theory, NAND circuits, circuit discovery, recursive dynamics, complex dynamics, fractal generation, cellular automata, emergent computation, concept formation, internal representations, mechanistic interpretability, automated discovery.

A collaboration with OpenAI GPT-5.5 Thinking. CC4.0. 143 pages and 63 references.

https://orcid.org/0009-0004-4206-4253

Outline

1. Introduction: From Preselected Objects to Object Formation

1.1. The conventional object-first structure of mathematics

1.2. Algebraic Abiogenesis as an operation-first alternative

1.3. The extension from algebraic identities to general generative systems

1.4. The hypothesis that learning and compression construct objects

1.5. Scope, claims, and organization of the paper

2. The General Architecture of Abiogenetic Discovery

2.1. Primitive substrates

2.2. Anonymous generators

2.3. Closure under composition, recursion, and local updating

2.4. Observation policies and behavioral equivalence

2.5. Stability, recurrence, and adversarial rejection

2.6. Quotients, canonical representatives, and emergent objects

2.7. A field-independent minimal toolbox

3. Algebraic Abiogenesis as the Foundational Case

3.1. Anonymous finite operations and formal term construction

3.2. Term-function collisions as observational evidence

3.3. Degeneracy and definitional-alias rejection

3.4. Explanation-subtracted fertility

3.5. Cross-order recurrence and theory ecology

3.6. Adversarial countermodels and recursive theory peeling

3.7. Fully invariant congruences and quotient term algebras

3.8. Free structures, normal forms, and decision procedures

3.9. What the algebraic calibration establishes

4. Logical Abiogenesis: From Anonymous Semantics to Consequence

4.1. Why equational collisions alone do not constitute logic

4.2. Anonymous finite logical matrices

4.3. Designated subsets without prior truth interpretation

4.4. Emergence of semantic consequence

4.5. Cross-order recurrence of logical fingerprints

4.6. Rejection of duplicated-value expansions

4.7. Reduced logical matrices and robust theory clusters

4.8. Post hoc recognition of conjunction-like and implication-like behavior

4.9. Limits of bounded logical discovery

5. From Consequence to Proof Calculus

5.1. Formula quotients under mutual consequence

5.2. Extraction of axioms and inference rules

5.3. Premise minimization and rule anti-unification

5.4. Introduction and elimination behavior

5.5. Modus ponens and implication adjunction

5.6. Countermodels for invalid generalizations

5.7. Bounded soundness and completeness

5.8. Rank-two closure and finite proof-theoretic certification

5.9. The path toward unbounded completeness and normalization

6. Circuit Abiogenesis from NAND Composition

6.1. NAND as a primitive generator rather than a named design language

6.2. Recursive circuit construction and truth-vector equivalence

6.3. Formula trees versus shared directed acyclic graphs

6.4. Exact discovery of two-input Boolean functions

6.5. Exact discovery of three-input Boolean functions

6.6. Minimal circuit complexity and certified lower bounds

6.7. Recurrent subcircuits and modular reuse

6.8. Emergence of XOR-like motifs

6.9. Multi-output conservation and arithmetic modules

6.10. Post hoc recognition of half-adders, full-adders, and multiplexers

6.11. Toward autonomous circuit invention

7. Recursive Abiogenesis: The Minimal Theory of Iterated Systems

7.1. Recursion as a distinct generative closure

7.2. Anonymous unary maps on finite carriers

7.3. Orbits, transient states, and periodic cycles

7.4. Attractors and basins as emergent objects

7.5. Eventual equivalence and orbit quotients

7.6. Recursive depth, stability, and structural recurrence

7.7. From finite iteration to continuous-state dynamics

8. Complex-Plane and Fractal Abiogenesis

8.1. The minimal substrate of complex points and transformations

8.2. Anonymous affine maps and transformation ecology

8.3. Why affine iteration alone rarely produces fractal boundaries

8.4. Nonlinearity as one source of recursive complexity

8.5. Branching as an alternative source of fractal structure

8.6. Iterated families of anonymous affine maps

8.7. Boundedness, occupancy, and scale-dependent observation

8.8. Multi-resolution persistence and dimension estimates

8.9. Rejecting collapse, divergence, and geometrically barren systems

8.10. Discovering fractal attractors without selecting known formulas

9. Cellular Automaton Abiogenesis

9.1. Cellular automata as spatially distributed recursion

9.2. Anonymous local update rules

9.3. Exhaustive binary radius-one rule ecologies

9.4. Fixed, periodic, chaotic, and propagating regimes

9.5. Translation-equivalence and moving-pattern objecthood

9.6. Localized structures as emergent computational modules

9.7. Pattern collisions and interaction laws

9.8. Conservation laws and reversibility

9.9. Compression, predictability, and computational fertility

9.10. From anonymous rules to emergent machines

10. Neural Abiogenesis: Training as Ground-Up Object Formation

10.1. Neural networks as recursively composed operational systems

10.2. Optimization as developmental recursion

10.3. Data distinctions and task-relevant equivalence

10.4. Functional substitutability of internal states

10.5. Activations as representatives of behavioral equivalence classes

10.6. The birth of internal objects during training

10.7. Features, concepts, and algorithms as quotient structures

10.8. Reusable neural circuits and modular computation

10.9. Recurrence across seeds, widths, and architectures

10.10. Neural objecthood as causal rather than geometric identity

11. Model Compression as Quotient Formation and Theory Peeling

11.1. Compression as removal of distinctions without behavioral loss

11.2. Pruning and parameter equivalence

11.3. Quantization and numerical quotienting

11.4. Low-rank factorization and redundant directions

11.5. Distillation and behavioral equivalence between models

11.6. Bottlenecks and forced abstraction

11.7. Compression as creation versus revelation of structure

11.8. Measuring which objects survive compression

11.9. Detecting the developmental birth point of an internal quotient

11.10. Compression as a microscope for mechanistic interpretability

12. A Unified Research Program for Minimal Toolboxes

12.1. Minimal field-specific substrates and generators

12.2. Algebra, logic, circuits, recursion, geometry, and automata

12.3. Shared criteria for stable object formation

12.4. Audit certificates and reproducible discovery pipelines

12.5. Distinguishing calibration, rediscovery, and genuine novelty

12.6. Cross-field transfer of discovered modules

12.7. Semantic and syntactic routes to object birth

12.8. Autonomous theorem, rule, circuit, and concept discovery

12.9. Experimental tests of Neural Abiogenesis

12.10. Mathematical objects and intelligent representations as stable functional quotients

12.11. Limitations and unresolved questions

12.12. Conclusion: Compression, equivalence, and the construction of worlds

13. References

1. Introduction: From Preselected Objects to Object Formation

Mathematics normally begins after a decisive conceptual act has already occurred. A domain has been specified, its elements have been distinguished, its operations have been named, and the identities governing those operations have been selected. The subsequent task is then to derive consequences. Group theory begins with a carrier equipped with an associative binary operation, identity, and inverses. Ring theory begins with two coordinated operations. Lattice theory begins with order-compatible meet and join. Logic begins with propositions, connectives, truth conditions, or proof rules. Circuit design begins with known gates and desired functions. Dynamical systems begin with a state space and an update law. Neural learning begins with an architecture, an objective, and a parameterized family of transformations.

These starting points are mathematically productive, but they conceal a prior question: how did the relevant objects, operations, distinctions, and laws become available for mathematics to study? In conventional practice, they are introduced by definition. The definition may be historically motivated, physically interpreted, or justified by previous examples, but it is nevertheless imposed from above. Once the definition is accepted, the object exists within the formal theory.

The present work investigates the reverse direction. Rather than beginning with a named mathematical species, it begins with primitive substrates, anonymous generators, recursive closure, and observational comparison. Structure is not assumed at the outset. It is inferred from recurrent patterns of equivalence, persistence, compression, and lawful substitution. Objects arise when multiple generated states or expressions become interchangeable under a specified observational policy. Laws arise when those equivalences remain stable across examples, scales, carrier sizes, transformations, and adversarial tests. The resulting program extends Algebraic Abiogenesis into logic, proof systems, circuit formation, recursion, complex dynamics, fractal generation, cellular automata, and neural representation learning.

The unifying claim is methodological. Mathematical and computational objects need not always be treated as primitive. They can be investigated as products of generative processes that progressively erase irrelevant distinctions while preserving behavior that matters. In this view, object formation is a quotient process, recursion is a source of candidate structure, and compression is a selection pressure favoring stable reusable forms.

1.1. The conventional object-first structure of mathematics

The dominant organization of mathematics is object-first. A theory begins by defining a class of entities and then studying the consequences of that definition. This procedure is so familiar that the creative act of object selection often disappears from view. Students encounter groups, vector spaces, fields, topological spaces, manifolds, graphs, categories, and logical calculi as

established structures. The existence of the class is taken for granted once its axioms have been written.

This approach has several advantages. It gives precise boundaries to a subject, permits rapid theorem development, and allows many examples to be treated within a common language. It also supports abstraction by separating essential structure from accidental presentation. A permutation group, a matrix group, and a symmetry group can be studied together because each instantiates the same axioms.

Yet this success creates a methodological asymmetry. Mathematics is highly developed at deriving consequences from selected axioms, but less systematic at explaining how the axioms themselves might be discovered from minimally interpreted data. Historically, algebraic structures often emerged through a long process of examples, failed generalizations, changing notation, and retrospective purification. Once stabilized, that historical process was compressed into a definition. The mature theory presents the final object but not the mechanism by which the object was formed.

A similar asymmetry appears in logic. Formal systems normally begin with a language, formation rules, axioms, and rules of inference, or with a class of semantic models. The task is then to establish soundness, completeness, decidability, normalization, or expressive power. What is rarely studied computationally is how a consequence relation or proof rule might arise from anonymous finite semantic behavior before any connective has been identified as implication, conjunction, or negation.

The same pattern recurs in circuit theory. A desired Boolean function is specified, and gates are composed to implement it. In dynamical systems, the map is selected before its orbits are analyzed. In fractal geometry, a recurrence or iterated function system is often chosen because its eventual form is already known to be interesting. In machine learning, an objective function determines what distinctions are rewarded or ignored, but the emergence of internal objects is usually described only after training has occurred.

The object-first approach therefore leaves open a foundational problem: can mathematics formalize the process by which objects become identifiable before their names and laws are known?

1.2. Algebraic Abiogenesis as an operation-first alternative

Algebraic Abiogenesis was introduced as an operation-first response to this problem. It begins not with groups, rings, lattices, or another named class, but with finite carriers and anonymous operation tables. Formal terms are composed from these operations, and terms are evaluated across the carriers. When two syntactically different terms induce the same term function, their collision is treated as observational evidence of an identity.

No particular identity is initially privileged. Associativity, commutativity, idempotence, cancellation, absorption, and distributivity are not supplied as targets. They may emerge, fail to emerge, or appear only as finite accidents. The principal task is therefore not merely to collect collisions, but to determine which collisions support a stable theory.

This requires progressive rejection. Constant operations, projections, duplicate presentations, and operations that ignore arguments can generate many identities while producing little mathematical structure. Small finite carriers can also satisfy accidental identities that disappear at larger orders. A large collision count may therefore reflect degeneracy rather than fertility. Algebraic Abiogenesis addresses this problem through explanation subtraction, cross-order recurrence, theory clustering, adversarial countermodels, consequence-reserve analysis, and recursive peeling.

The surviving identities are not merely descriptive regularities. They define congruences on formal term algebras. Expressions that are observationally indistinguishable are identified, and the quotient becomes a new object. Canonical representatives can then be sought, rewrite rules can be oriented, free structures can be examined, and decision procedures can emerge.

The central reversal is thus:

conventional algebra proceeds from axioms to models and consequences;

Algebraic Abiogenesis proceeds from anonymous models to recurrent identities, quotient objects, and inferred theories.

The term abiogenesis is used methodologically rather than biologically. It indicates the formation of organized mathematical structure from a substrate that does not initially contain the named species later recognized within it.

1.3. The extension from algebraic identities to general generative systems

The present paper broadens this methodology beyond equational algebra. The key observation is that the algebraic experiment contains a more general architecture:

a primitive substrate is supplied;

anonymous generators act upon it;

closure produces a larger ecology of expressions, states, or transformations;

an observation policy determines which outcomes count as equivalent;

stability filters reject accidental, degenerate, or nonpersistent structure;

the surviving equivalence classes become objects.

In logic, the substrate consists of finite values and designated subsets. Anonymous operations generate formulas. Preservation of designation induces semantic consequence. Mutual consequence defines formula equivalence classes, and stable patterns of consequence can generate axioms and inference rules.

In circuits, input truth functions form the substrate. NAND composition generates larger circuit states. Circuits with identical truth vectors are identified. Exact minimization and shared-subcircuit analysis reveal reusable modules, including arithmetic and routing structures recognized only after their behavior has emerged.

In recursive systems, anonymous update maps generate orbits. States become equivalent when they share eventual behavior, enter the same attractor, or support the same recurrent pattern. In the complex plane, anonymous transformations and branching recursions can generate scale-persistent geometry. In cellular automata, local rules recursively generate space-time patterns, and translation-equivalence can convert moving configurations into stable objects.

These fields differ in content but share a common construction. The candidate objects do not exist as primitive named entities. They emerge because a generative process repeatedly produces outcomes that can be compressed into stable behavioral classes.

1.4. The hypothesis that learning and compression construct objects

The most consequential extension concerns artificial intelligence. Neural networks are normally described as parameterized functions trained to reduce predictive error. Their internal representations are then interpreted as features, concepts, directions, circuits, or algorithms. This language assumes that meaningful internal objects have somehow formed, but the mechanism of their formation is not yet fully understood.

The hypothesis developed here is that training under representational constraints performs a process analogous to quotient formation. The model is exposed to many distinct inputs, states, and contexts. To predict successfully with finite capacity, it cannot preserve every difference among them. It must retain distinctions that affect future behavior and suppress those that do not.

Two internal states may be numerically different yet functionally interchangeable if replacing one with the other leaves downstream predictions unchanged. Such states belong to the same task-relative equivalence class. A feature or concept may therefore be understood not as a particular neuron or vector, but as a stable class of internal states that produce equivalent consequences under the model's computational policy.

Compression intensifies this process. Pruning removes parameters whose contribution is behaviorally redundant. Quantization identifies nearby numerical states. Low-rank factorization

collapses redundant directions. Distillation replaces one implementation with another that preserves output behavior. Bottlenecks force many distinguishable inputs into fewer latent states. In each case, distinctions are removed while selected function is retained.

This suggests two possibilities. Compression may create structure by forcing the model to invent reusable abstractions, or it may reveal structure that training had already formed by removing redundant implementation detail. These possibilities are not mutually exclusive. Continuous capacity pressure may create quotients during training, while post-training compression may expose their minimal realization.

Neural Abiogenesis therefore asks whether the birth of a feature, concept, or circuit can be detected as the stabilization of a functional equivalence relation during learning.

1.5. Scope, claims, and organization of the paper

This paper develops a unified framework rather than claiming that every field has already been solved by a single algorithm. The strongest completed results lie in finite equational theory formation, logical consequence extraction, bounded proof-calculus construction, and exact NAND-circuit discovery. The extensions to recursive dynamics, complex-plane systems, fractals, cellular automata, and neural training are presented as a coordinated research program with increasingly ambitious experimental requirements.

The principal claims are fourfold.

First, object formation can be treated mathematically as the stabilization of equivalence under a generative and observational process.

Second, the same architecture applies across fields whose surface objects appear very different.

Third, recursion and composition generate candidate structure, while compression, recurrence, and adversarial rejection determine which structures deserve object status.

Fourth, neural representation learning may be interpreted as an internal form of the same process, in which task-relevant distinctions survive and irrelevant distinctions are quotiented away.

Sections 2 and 3 formalize the general architecture and its algebraic foundation. Sections 4 and 5 extend the method from equational identity to logical consequence and proof calculus. Section 6 develops circuit formation through anonymous NAND composition and exact shared-subcircuit discovery. Sections 7 through 9 examine recursion, complex and fractal construction, and cellular automata as additional generative domains. Sections 10 and 11 develop Neural Abiogenesis and model compression as functional quotient formation. Section 12 unifies these results into a program of minimal field-specific toolboxes and reproducible discovery systems.

The broader aim is not to eliminate definitions from mathematics. It is to supplement definition-first mathematics with a discovery theory of how definitions, objects, and laws can arise. The question is no longer only what follows once an object has been named. It is also how stable objects become available for naming at all.

2. The General Architecture of Abiogenetic Discovery

Abiogenetic discovery begins before a field has stabilized its preferred objects. It does not assume that the correct entities, operations, invariants, or rules have already been identified. Instead, it specifies a minimal generative environment and asks which structures become stable under repeated construction and observation. The resulting method is neither pure enumeration nor ordinary induction from examples. It is a controlled process of generation, comparison, rejection, compression, and quotient formation.

Across algebra, logic, circuits, recursion, complex dynamics, cellular automata, and neural systems, the same architecture recurs. A substrate provides primitive states. Anonymous generators transform or combine those states. Closure under composition, recursion, or local updating produces a rapidly expanding space of candidates. An observation policy determines which generated outcomes count as equivalent. Stability filters reject structures that are degenerate, accidental, scale-dependent, presentation-dependent, or mathematically barren. The remaining equivalence classes become candidate objects, while recurrent relations among them become candidate laws.

The architecture can be summarized schematically as:

primitive substrate
→ anonymous generation
→ recursive or compositional closure
→ observational equivalence
→ stability filtering
→ quotient formation
→ canonical objects and reusable structure

The force of the method lies in the separation between generation and interpretation. The system is permitted to create and compare structures before familiar names are applied. Recognition occurs afterward. This avoids turning discovery into disguised confirmation of a known target.

2.1. Primitive substrates

Every abiogenetic system begins with a substrate. The substrate is not yet the mature subject matter of the field. It is the smallest domain on which candidate structure can be generated and observed.

In algebra, the substrate is a finite carrier. Its elements are initially anonymous. They need not be numbers, symbols, physical states, or semantic values. They are merely distinguishable positions on which operations can act.

In logic, the substrate may also be a finite carrier, but logical interpretation requires additional structure. A subset of values is marked as designated, although the values themselves are not initially called true, false, both, neither, necessary, or possible. The designated set supports preservation-based consequence without presupposing a known logical system.

In circuit discovery, the substrate consists of elementary input signals represented by truth functions. These inputs are not yet embedded in a designed circuit architecture. They are primitive behavioral sources from which larger computations can be composed.

In recursive dynamics, the substrate is a state space. It may be a finite set, a lattice, the real line, the complex plane, or a space of configurations. The fundamental question is what stable orbit structures arise under repeated application of anonymous update rules.

In cellular automata, the substrate is distributed. It consists of cells, local states, and a neighborhood relation. The global configuration is not updated by a single central operation but by simultaneous local transformations.

In neural systems, the substrate is more complicated. It includes data examples, parameters, activations, and computational paths. None of these alone should automatically be treated as the true object of the learned system. The relevant objects may instead be equivalence classes distributed across many activations and parameters.

A minimal substrate should satisfy two competing requirements. It must be simple enough that generation and comparison remain computationally tractable, yet rich enough to permit nontrivial distinctions. A substrate with too few states may support only degeneracy. A substrate with too many unconstrained dimensions may produce an intractable candidate space before stable structure can be identified.

The choice of substrate is therefore methodological rather than merely technical. It defines what forms of objecthood are even available to emerge.

2.2. Anonymous generators

A generator is any primitive rule that transforms, combines, or updates elements of the substrate. Generators are called anonymous when their familiar interpretation is deliberately withheld during discovery.

In finite algebra, a generator may be a binary or ternary operation table. It is not called multiplication, conjunction, implication, meet, join, or projection. The table is evaluated only by its action.

In logic, anonymous operations combine values and formulas. A truth table may later be recognized as conjunction-like or implication-like, but that label is not used to guide the search.

In circuit discovery, NAND is not anonymous in the historical sense because its behavior is supplied, but its compositions can still be treated without target functions. The discovery system is not instructed to build XOR, an adder, a selector, or parity. It generates circuits and groups them by behavior. The emergent modules are named only after their truth functions and reuse patterns have been observed.

In complex dynamics, an anonymous generator may be an affine or nonlinear map. The system need not be told that a map is a rotation, contraction, translation, or inversion. Those categories can be inferred from recurrent action on points and geometric relations.

In cellular automata, the local update rule is an anonymous generator. A rule is not initially labeled ordered, chaotic, conservative, universal, or glider-supporting. It is observed through the structures its iteration produces.

Anonymity is essential because target naming can smuggle the desired conclusion into the experiment. If the system is asked to find implication, it may merely optimize toward a known semantic signature. If it is asked to find a full-adder, it may search only circuits satisfying a supplied arithmetic relation. Calibration experiments may legitimately use known targets, but genuine abiogenetic discovery requires a phase in which candidate structure is generated independently of retrospective interpretation.

The generator alphabet should also be minimal. Too many primitive operations can make apparent structure trivial because the desired behavior is already available directly. A stronger result is obtained when complex structure emerges from a small set of primitive actions.

2.3. Closure under composition, recursion, and local updating

Generators alone do not produce a mature structure. The candidate universe emerges through closure.

In algebra, closure is term composition. Variables and operation symbols are recursively combined into formal expressions. Each new layer of terms permits more complex identities and more discriminating term functions.

In logic, closure is formula construction. Anonymous connectives are recursively applied to variables and previously formed formulas. Consequence is then evaluated over the resulting formula universe.

In circuits, closure is directed acyclic composition. New gates act on primary inputs or previously generated signals. Shared subcircuits create a distinction between formula trees and computational graphs.

In ordinary recursion, closure is temporal:

$x_{n+1} = f(x_n)$.

A single seed generates an orbit. Multiple seeds generate an orbit ecology.

In branching systems, closure has the form:

$S_{n+1} = \bigcup_j f_j(S_n)$.

Each state or set gives rise to multiple descendants. This is the simplest route by which affine transformations can generate fractal geometry.

In cellular automata, closure is synchronous local updating. The same local rule is repeatedly applied across a lattice, creating global behavior that is not explicitly encoded as a global object.

In neural networks, two closures operate simultaneously. Forward computation composes layers, while optimization recursively modifies the parameters that define those layers. The system that processes the next example is altered by the preceding training step.

Closure creates combinatorial expansion. The number of terms, formulas, circuits, or trajectories can grow exponentially or faster. This growth is not merely a computational nuisance. It is the source of candidate novelty. Stable objects can emerge only because many distinct generative histories converge to the same behavior.

The challenge is therefore to preserve enough closure to allow structure while preventing the candidate universe from becoming unmanageable. Typical strategies include bounded depth,

canonical syntactic generation, semantic deduplication, symmetry reduction, incremental expansion, and compression by behavioral equivalence.

2.4. Observation policies and behavioral equivalence

Generation produces possibilities, but observation determines identity. An observation policy specifies which differences matter and which may be ignored.

In algebra, two terms are observationally equivalent when they induce the same term function on a selected family of finite algebras.

In logic, two formulas may be equivalent when they preserve designation in the same contexts, or when each semantically entails the other.

In circuits, two circuits are equivalent when they compute the same truth function, even if their internal gate graphs differ.

In recursion, two states may be equivalent when they eventually enter the same cycle, share the same attractor, or have the same symbolic orbit signature.

In geometry, two generated point sets may be equivalent under translation, rotation, scaling, or another selected symmetry.

In cellular automata, two patterns may represent the same moving object when one evolves into a translated copy of the other after a fixed number of time steps.

In neural systems, two internal states may be equivalent when substituting one for the other leaves downstream behavior unchanged across a specified family of contexts.

The observation policy is not a neutral afterthought. It determines the ontology of the resulting theory. Exact numerical equality yields a different object system from task-relative substitutability. Equality under all contexts is stricter than equality under a limited test family. Translation equivalence produces moving cellular objects, whereas pointwise equality does not.

The policy must therefore be explicit and auditable. Hidden observation choices can generate hidden assumptions. A credible abiogenetic experiment must state:

what is observed;

over which contexts;

at what resolution;

under which transformations;

with what tolerance;

and with what evidence of stability.

Object formation is always relative to some observational policy, but this relativity need not make the result arbitrary. Policies can themselves be compared by explanatory power, compression, robustness, and predictive success.

2.5. Stability, recurrence, and adversarial rejection

A single observed equivalence is not yet a law. Finite systems generate coincidences. Small carriers satisfy accidental identities. Numerical discretization produces false stability. Symmetry duplicates may inflate recurrence. Highly degenerate generators can create enormous numbers of uninformative collisions.

Abiogenetic discovery therefore requires rejection mechanisms.

The first filter is degeneracy. Constant maps, projections, ignored arguments, duplicated generators, and trivial fixed structures must be identified before fertility is assessed.

The second filter is explanation subtraction. If a simple known or previously discovered rule explains a large family of observations, those consequences should be removed before residual fertility is measured. Otherwise the system may repeatedly rediscover the same structure in different syntactic forms.

The third filter is recurrence. A candidate law or object should appear across independently generated examples, different carrier sizes, different seeds, or different levels of scale. Cross-order recurrence is especially important in finite algebra and logic because it separates theory from small-model accident.

The fourth filter is adversarial testing. A candidate explanation should be challenged by models constructed or selected to satisfy its apparent basis while violating additional residual claims. Countermodels reveal whether a proposed rule genuinely explains the observations or merely fits a narrow sample.

The fifth filter is perturbational robustness. In continuous and neural systems, small changes in parameters, inputs, or architecture should not immediately destroy a supposedly stable object unless fragility is itself part of the structure.

The sixth filter is consequence reserve. A fertile structure should support nontrivial further development. A theory that compresses many observations but yields no new predictions, normal forms, modules, or derived laws may be descriptively concise yet mathematically barren.

The seventh filter is recursive peeling. Once a dominant rule is identified, its consequences are removed and the residual is analyzed again. This permits layered theory formation rather than forcing all observed structure into a single explanation.

These filters transform raw recurrence into certified structure.

2.6. Quotients, canonical representatives, and emergent objects

The decisive mathematical step occurs when observational equivalence is promoted into a quotient.

Suppose a generated universe G is equipped with an equivalence relation $\sim$. The quotient $G/\sim$ replaces each collection of mutually equivalent presentations with a single class. The class becomes the object.

In algebra, the quotient of the term algebra by a fully invariant congruence produces the free algebra of the discovered theory.

In logic, quotienting formulas by mutual consequence produces a Lindenbaum-like structure whose elements are inferential classes rather than strings.

In circuits, quotienting circuit graphs by truth-function equivalence produces the space of computable Boolean functions. Minimal circuits can then serve as canonical representatives.

In recursion, quotienting states by eventual behavior produces basins, cycles, or attractor classes.

In cellular automata, quotienting patterns by translation through time produces persistent moving objects.

In neural systems, quotienting internal states by downstream substitutability would produce functional concepts or computational roles.

A quotient alone may still have many equivalent representations. Canonicalization selects preferred representatives. These may be shortest terms, normal forms, minimal circuits, compressed state descriptions, or low-complexity neural modules.

Canonical representatives are important because they turn equivalence into usable computation. They support decision procedures, retrieval, reuse, and comparison. A normal form answers whether two expressions are equal. A minimal netlist reveals shared structure. A compressed neural circuit may expose the causal mechanism underlying a distributed representation.

Objecthood therefore has three levels:

a set of presentations;

an equivalence relation among them;

a canonical or reusable representative of each class.

The object is not identical to any one presentation. It is the stable class that survives changes of presentation.

2.7. A field-independent minimal toolbox

The general methodology can be summarized as a minimal seven-part toolbox.

First, define a substrate. Specify the primitive states without prematurely assigning mature semantic categories.

Second, define anonymous generators. Supply the smallest operations or update rules capable of producing a nontrivial candidate space.

Third, generate closure. Compose, iterate, branch, or locally update the substrate.

Fourth, define an observation policy. State precisely when two generated outcomes count as behaviorally equivalent.

Fifth, filter for stability. Reject degeneracy, finite accidents, duplicate presentations, nonrecurrent patterns, and fragile explanations.

Sixth, form quotients. Treat stable equivalence classes as candidate objects.

Seventh, test fertility. Determine whether the objects support canonical forms, laws, modules, predictions, proofs, or further generative structure.

This toolbox is field-independent, but it does not make every field identical. The substrate and observation policy remain domain-sensitive. What counts as equivalence in proof theory differs from what counts as equivalence in geometry or neural computation. The unification lies not in erasing these differences but in showing that each field can be studied through the same sequence of generative closure, observational identification, and stable quotient formation.

The deeper proposal is that object formation is not an isolated event at the beginning of mathematics. It is a recurring process wherever many presentations, trajectories, or computations are compressed into stable functional classes. Algebraic laws, logical propositions, circuit modules, attractors, fractals, cellular objects, and neural concepts may all be instances of the same general phenomenon: structure becomes an object when recursive generation repeatedly produces distinctions that can be safely forgotten.

3. Algebraic Abiogenesis as the Foundational Case

Algebraic Abiogenesis provides the foundational case because equational algebra offers a particularly clean environment in which object formation can be separated from prior interpretation. The substrate is finite, the operations can be represented exactly, formal expressions can be generated recursively, and equality can be tested without approximation. This makes the field suitable for a rigorous demonstration of the larger methodological claim: stable mathematical structure can arise from anonymous operational information before a named algebraic species has been selected.

The method begins with finite carriers and operation tables, not with groups, rings, lattices, quasigroups, or other established classes. Terms are formed from variables and anonymous operation symbols. Their induced functions are compared across finite models. Exact collisions are treated as observations, but not immediately as laws. A sequence of filters separates triviality, definitional redundancy, local accidents, and mathematically barren regularities from identities that recur across orders and support further consequences.

The surviving identities generate congruences on free term algebras. Quotienting by those congruences produces new algebraic objects, while canonical forms and rewrite systems provide effective decision procedures. Algebraic Abiogenesis is therefore not merely a search for equations. It is a process by which operational data is compressed into a stable theory whose objects, laws, and computational procedures were not supplied at the beginning.

3.1. Anonymous finite operations and formal term construction

Let X be a finite carrier of order n. An anonymous operation of arity k is a function

$f: X^k \to X$.

The operation is specified only by its table. No semantic interpretation is attached to its entries. A binary table is not initially regarded as addition, multiplication, implication, meet, or composition. A ternary table is not treated as a discriminator, majority operation, conditional, or affine combination. It is merely an exact rule assigning an output to every input tuple.

A signature Σ consists of one or more anonymous operation symbols with fixed arities. Given a set of variables V, the free term algebra TΣ(V) is generated recursively. Every variable is a term. If f is a k-ary operation symbol and $t_1, \ldots, t_k$ are terms, then

$f(t_1, \ldots, t_k)$

is also a term.

This recursive grammar generates a rapidly expanding universe. Even a single binary operation produces variables, depth-one terms, nested terms, repeated-variable terms, balanced trees,

and highly asymmetric compositions. The syntactic term count grows much faster than the number of distinct functions that can be realized on a small finite carrier. Consequently, collisions are inevitable.

The experimental challenge is to construct a term universe large enough to reveal structure while remaining computationally tractable. Bounded operation count, bounded depth, canonical variable ordering, syntax deduplication, and semantic hashing are useful controls. Importantly, term generation must not privilege familiar equations. Associative, commutative, or idempotent patterns should arise only because they appear within the general recursive grammar.

Each anonymous operation table interprets every term as a concrete function

$t^A: X^r \to X$,

where r is the number of variables under consideration and A denotes the finite algebra. The resulting term functions constitute the primitive observational data of Algebraic Abiogenesis.

3.2. Term-function collisions as observational evidence

Two formal terms s and t collide on a finite algebra A when they induce exactly the same term function:

$s^A = t^A$.

Equivalently, for every valuation of the variables into X,

$s^A(v) = t^A(v)$.

A collision is therefore stronger than agreement on a few sampled inputs. It is an exact equality over the entire finite valuation space.

Such a collision can be written provisionally as the identity

$s \approx t$.

At this stage, however, the expression is observational rather than axiomatic. It records what happened in one or more finite models. The system does not yet know whether the equality reflects a general law, a consequence of a simpler law, a degeneracy of the operation, or a coincidence caused by small carrier size.

Term-function collisions are useful because they transform raw tables into relations among expressions. A table may contain no obvious conceptual structure when inspected entry by entry, yet its induced term functions may reveal extensive regularity. Distinct syntax trees can collapse onto the same function, generating equivalence classes of expressions.

For a finite carrier of order n and r variables, at most $n^{(n^r)}$ distinct r-ary functions exist. The formal term algebra, by contrast, is infinite whenever the signature contains a positive-arity operation. As terms grow, repeated functional collisions are unavoidable. The significant question is not whether collisions occur but how they are organized.

A provisional equational theory for A can be defined as

$Eq(A) = \{(s,t) : s^A = t^A\}$.

For a family of algebras C, the common observational theory is

$Eq(C) = \bigcap_{a \in C} Eq(A)$.

This intersection is the first candidate for stable structure. Identities that survive across independently generated algebras are less likely to be artifacts of one table. Yet recurrence alone is insufficient unless degeneracy and common low-information explanations are removed.

3.3. Degeneracy and definitional-alias rejection

Anonymous operation spaces contain many structures that generate large numbers of identities for trivial reasons. Without explicit rejection, these operations dominate collision statistics and can falsely appear mathematically fertile.

A constant operation ignores all of its inputs. Every term containing it may collapse to one value, producing extensive but uninformative equivalence. A projection returns one selected coordinate and ignores the others. Such an operation can create many simple identities while contributing little internal structure. More generally, an operation may depend essentially on only a proper subset of its coordinates.

Essential-variable testing therefore forms an early filter. For each input position, the algorithm asks whether changing that coordinate while holding the others fixed can alter the output. Operations that fail essential dependence requirements can be rejected or classified separately.

Another source of inflation is duplication under carrier relabeling. Two operation tables related by a permutation of X represent isomorphic finite algebras. Treating them as independent evidence would overcount one structure. Canonicalization under the action of the symmetric group, or explicit orbit reduction, prevents this duplication.

Multi-operation signatures introduce definitional aliases. One anonymous operation may be a short term in the others. Two symbols may also represent identical tables or variants related by argument permutation. If such aliases are not detected, the system may mistake notational redundancy for algebraic richness.

Short-term definability tests can identify whether an operation symbol is reproduced by a low-complexity term over the remaining signature. Duplicate operations, coordinate permutations, and simple derived operations should be recorded as explanatory relations rather than counted as independent generators.

Degeneracy rejection does not imply that constant, projective, or reducible systems have no mathematical interest. It means only that their collision abundance should not be confused with the emergence of a deep equational theory. They may form separate ecological classes with their own characteristic behavior.

3.4. Explanation-subtracted fertility

Raw identity count is a poor measure of theoretical value. One simple law may generate hundreds or thousands of bounded term collisions. If all such consequences are counted independently, a theory can appear highly fertile even though its observations are explained by a single elementary relation.

Explanation-subtracted fertility addresses this problem. Suppose a candidate identity basis B has been identified. Let C be the observed collision set in a bounded term universe. The identities derivable from B within the chosen proof or normalization procedure are removed:

$R = C \setminus \mathrm{Conseq}(B)$.

The residual R measures what remains unexplained.

For example, a diagonal identity of the form

$m(x,x,x) \approx x$

may explain many collisions involving repeated variables and nested diagonal terms. Once those consequences are subtracted, a previously large collision family may have no remaining content. Such a cohort is locally collision-rich but theoretically exhausted.

By contrast, if a substantial residual remains after subtraction, the model family may contain additional structure. The system can then search the simplest residual identities for a second explanatory rule. After adding that rule to B, subtraction is repeated. This creates a layered analysis:

observations
→ simplest explanation
→ explained consequences removed
→ residual theory
→ next explanation.

Fertility should therefore be measured not by the total number of collisions but by the amount of coherent, compressible structure that survives known explanations. A useful theory has consequence reserve: it supports further nonredundant identities, classifications, free structures, or decision procedures after its obvious regularities have been accounted for.

This principle parallels scientific model selection. A theory that merely restates its observations has little explanatory compression. A compact basis that accounts for many independent collisions is more significant.

3.5. Cross-order recurrence and theory ecology

Finite identities can be accidental. An equation may hold on all algebras of a small order yet fail immediately at the next order. Cross-order recurrence is therefore a central filter.

Instead of studying isolated operation tables, Algebraic Abiogenesis constructs a theory ecology. Each finite algebra is represented by an identity fingerprint over a common term universe. Similarity or overlap among fingerprints defines a graph or weighted network. Clusters correspond to families of operations sharing substantial equational behavior.

A cross-order cluster contains algebras on carriers of different sizes. Such recurrence is stronger than repetition among relabeled copies of one finite structure. It suggests that the observed identities reflect a genuine variety-like law rather than a peculiarity of one cardinality.

The ecology may contain several distinct regimes:

collision-rich degenerate models;

models sharing only a simple diagonal law;

locally fertile models with no cross-order continuation;

coherent clusters supporting a deeper basis;

isolated structures with unusual but nonrecurrent identities.

Cluster selection must be handled carefully. A fixed raw-overlap threshold can behave inconsistently when term-universe size changes. Normalized measures, stability across resampling, and persistence over thresholds provide stronger evidence. The recovered law should be more stable than any particular clustering heuristic.

The purpose of the ecology is not to force all operations into one classification. It is to permit multiple provisional theories to coexist, compete, merge, split, or disappear as the observational universe expands.

3.6. Adversarial countermodels and recursive theory peeling

Cross-order recurrence reduces accidental structure but does not eliminate it. Candidate laws must be tested against adversarial models.

Suppose a cluster appears to be explained by a basis B, yet additional identities remain common to the sampled models. To determine whether those identities follow from B, the system searches for a finite algebra satisfying B while violating a residual identity.

If such a countermodel exists, the residual identity was accidental to the original sample. Adding the countermodel to the cohort removes the false regularity from the intersection. If repeated countermodel search fails and the residual persists across expanding orders and term universes, confidence in a genuine consequence increases.

This process can be recursive. After one layer of accidental identities is removed, a new residual pattern may become visible. Theory peeling alternates among:

candidate-basis extraction;

consequence subtraction;

countermodel generation;

cohort enlargement;

residual recomputation.

The goal is not necessarily to find one final axiom immediately. It is to create an audit trail showing why each observed identity survives.

Adversarial testing also protects against overgeneralization. A candidate rewrite rule inferred from several finite examples may fail under a slightly different table. Countermodels expose the boundary of its validity and help separate a law from a statistical resemblance.

A mature birth certificate for an algebraic theory should therefore include not only supporting models but also rejected candidates and explicit separating countermodels.

3.7. Fully invariant congruences and quotient term algebras

The transition from observations to objects occurs through congruence formation.

For any algebra A, evaluation of terms defines a homomorphism from the free term algebra into the algebra of term functions on A. The kernel of this evaluation map identifies terms having the same interpretation. Kernels of homomorphisms are congruences.

For a family C of algebras, the intersection of these kernels remains a congruence. Because the relation is preserved under substitution of terms for variables, it is fully invariant. Thus the observational equivalence generated by common term-function collisions is not merely an arbitrary clustering of expressions. It is compatible with every operation and every endomorphism of the free term algebra.

Let θC denote this equivalence. The quotient

$$T\Sigma(V)/\theta C$$

is an algebra whose elements are equivalence classes of terms. Two expressions represent the same quotient element exactly when the selected observational family cannot distinguish them.

This quotient is the canonical object produced by the experiment. Its elements were not specified beforehand. They arise from the collapse of syntactic presentations under stable operational equivalence.

The quotient may be finite or infinite. It may exhibit free growth, collapse to a small structure, contain towers of irreducible terms, or admit a simple normal-form grammar. These properties determine whether the discovered theory is mathematically substantial.

The fully invariant nature of θC also explains why discovered identities support lawful substitution. If $s \approx t$ holds in the quotient, then replacing variables by arbitrary terms preserves the equality. The resulting object is therefore a genuine equational theory rather than a list of isolated empirical coincidences.

3.8. Free structures, normal forms, and decision procedures

Once a candidate identity basis has been extracted, the next task is to determine its computational and structural consequences.

A rewrite rule or set of rules can orient identities toward simpler representatives. If rewriting terminates and is confluent, every term has a unique normal form. Equality in the quotient is then decidable by normalization:

$$s \approx t$$

if and only if

$$NF(s) = NF(t).$$

Even when a general confluence proof is unavailable, bounded exhaustive tests can verify consistency over large term universes and compare alternative reduction strategies. Agreement among outermost, innermost, and randomized reductions provides evidence for canonical behavior.

Free algebras offer another measure of fertility. The free algebra on one generator may collapse to a singleton, while the free algebra on two generators may remain infinite. Such rank-dependent behavior reveals structural richness that is not visible from the original finite tables alone.

Distinct irreducible towers can witness unbounded growth. Enumeration of finite models can test formulas for the number of labeled algebras at each order. Canonical representatives permit exact counting, efficient comparison, and generation of new examples.

The emergence of a decision procedure is especially significant. The initial input consisted only of finite tables and term evaluations. After theory formation, the system can decide equality for expressions not present in the original sample. This is a clear transition from data collection to mathematical generalization.

3.9. What the algebraic calibration establishes

The algebraic calibration establishes a bounded but concrete form of ground-up theory formation.

It shows that anonymous finite operations can generate exact term-function collisions; that trivial and accidental structure can be progressively removed; that recurrent identity families can be isolated across carrier orders; and that compact explanatory laws can account for large collision sets. It further shows that the surviving identities define fully invariant congruences, quotient term algebras, free structures, normal forms, and word-problem procedures.

The calibration does not show that every run produces a novel algebraic species. Rediscovery of a known law remains valuable as a validation of the method, but it must not be confused with mathematical novelty. Nor does the current implementation prove that every stage of theory rejection and maturity assessment has been fully automated for every signature.

The central result is methodological rather than classificatory:

finite operational information can be transformed into stable equational objecthood without beginning from a named algebraic class or preferred identity.

This foundational case supplies the template for the rest of the paper. Logic replaces term equality with preservation-based consequence. Circuit theory replaces terms with shared computational graphs. Recursion replaces formal composition with temporal iteration. Geometry introduces scale and spatial observation. Neural learning replaces exact finite equality with task-relative functional substitutability.

Across these extensions, the same question persists: when do many generated presentations become one stable object?

4. Logical Abiogenesis: From Anonymous Semantics to Consequence

Logical Abiogenesis extends the operation-first methodology from equational algebra into logic by adding one ingredient that algebra alone does not contain: designatedness. An algebraic identity asks whether two terms always evaluate to the same element. A logical consequence relation asks whether the acceptance of some formulas guarantees the acceptance of another. Equality of term functions is therefore insufficient to generate logic. A logical system requires a distinction between values that count as accepted and values that do not.

The minimal semantic object is a finite logical matrix consisting of a finite carrier, one or more anonymous operations, and a nonempty proper designated subset. No carrier element is initially called true, false, both, neither, necessary, possible, or indeterminate. No operation is initially called conjunction, disjunction, implication, negation, or equivalence. The system sees only operation tables, designated subsets, formula evaluations, and preservation of designation.

From these primitives, semantic consequence can be defined without presupposing a familiar logic. Formula identities may still occur, but consequence is a richer observational relation. Two formulas can differ as term functions while remaining interchangeable with respect to designatedness. Conversely, formulas that agree on a small set of valuations may differ in inferential behavior once embedded in larger contexts.

The central computational question is whether stable logical theories recur across independently generated matrices and different carrier sizes. A positive result requires more than matching a few truth tables. Apparent cross-order recurrence can be produced by duplicating a value in a smaller matrix. Reducedness tests, enlarged formula universes, additional variables, substitution checks, and countermodels are therefore necessary before a recurrent logical fingerprint can be regarded as substantive.

The initial experiments show that anonymous finite matrices can generate stable consequence relations, mutual-entailment quotients, and bounded proof calculi. Several recurrent clusters display conjunction-like, implication-like, and reversed-implication-like behavior only after their semantic structures have been discovered. These results establish a bridge from Algebraic Abiogenesis to proof theory.

4.1. Why equational collisions alone do not constitute logic

An equational theory classifies expressions by equality of value. Logic classifies formulas by validity, consequence, incompatibility, and inferential role. These structures overlap, but they are not identical.

Suppose two formulas A and B induce the same truth function on every valuation. Then they are semantically equivalent in the strongest possible sense. They have the same value everywhere and are therefore interchangeable under any operation that respects equality.

However, logical equivalence can be weaker. If a matrix has several designated values, A and B may evaluate differently while always agreeing on whether their values are designated. In that case they are inferentially indistinguishable even though they are not equal as term functions.

For example, a three-valued logic may contain two distinct designated values. One formula may evaluate to the first designated value while another evaluates to the second. Equational analysis treats them as different. Consequence analysis may treat them as equivalent because both are accepted in exactly the same circumstances.

This distinction is essential. If Logical Abiogenesis were based only on term-function collisions, it would remain a branch of universal algebra applied to logical operations. It would discover identities among connectives, but not validity or inference.

A logic requires a preservation criterion. Given a designated set D, the semantic consequence relation is defined by:

$\Gamma \models A$

when every valuation that sends each formula in Γ to a value in D also sends A to a value in D.

This definition does not require the values in D to be called true. It requires only that designatedness be preserved.

The shift from equality to designatedness changes the ontology. In Algebraic Abiogenesis, the main quotient identifies terms that compute the same function across a model family. In Logical Abiogenesis, the primary quotient may identify formulas that entail one another, even when their exact semantic values differ.

A further distinction concerns premises. Equational identities are symmetric. If $s \approx t$, then $t \approx s$. Consequence is directional. Γ may entail A even when A does not entail every member of Γ. This asymmetry is what makes inference possible.

Logical Abiogenesis must therefore analyze at least three structures:

formula validity;

directional consequence;

mutual consequence.

Validity identifies formulas accepted under every valuation. Directional consequence identifies inferential dependencies. Mutual consequence identifies formulas occupying the same inferential role.

Equational collisions remain useful, but they become one layer within a richer semantic system.

4.2. Anonymous finite logical matrices

A finite logical matrix can be written as

M = (A, F, D),

where A is a finite carrier, F is a collection of operations on A, and D is a nonempty proper subset of A.

In the simplest experiment, F contains one anonymous binary operation b. A formula is generated from variables by recursive application of b:

x, y, z, b(x,y), b(y,x), b(x,b(y,z)), and so forth.

The operation table is not interpreted. For an n-element carrier, a binary operation is simply one of the $n^{(n^2)}$ possible functions from A^2 to A.

For a two-element carrier, there are $2^4 = 16$ binary operation tables.

For a three-element carrier, there are $3^9 = 19{,}683$ binary operation tables.

Each operation can be paired with every nonempty proper designated subset. A two-element carrier has two such subsets. A three-element carrier has six.

The resulting search space contains logical matrices rather than merely algebras. Each matrix combines operational behavior with an acceptance policy.

Not every matrix is informative. Some operations are constant. Some ignore one argument. Some differ only by relabeling the carrier. Some designated sets collapse most semantic distinctions. These cases must be classified and, where appropriate, rejected.

The anonymity requirement is important. It prevents the experiment from becoming a disguised search for classical implication, conjunction, or another selected connective. The program does not begin with a library of familiar truth tables. It enumerates the complete operation space at small orders and records the consequence structures that arise.

The formula universe must also be generated without privileging known logical forms. Variables are combined systematically up to a bounded operation count or depth. Duplicate syntax trees are removed. Carrier relabeling does not alter the logical fingerprint.

The output of each matrix is not a name but a data structure recording which formulas are valid and which finite premise sets entail which conclusions.

4.3. Designated subsets without prior truth interpretation

The designated subset is the minimal semantic device that turns an algebra into a logic.

If D contains a value a, the experiment does not assume that a means truth. It records only that formulas evaluating to a count as accepted for the purpose of consequence.

For a three-element carrier, the designated subset may contain one value or two values. These alternatives can produce substantially different logics even when the operation table remains unchanged.

This policy-relative character is not a defect. It exposes an important fact: logical identity depends not only on how formulas are evaluated but on which distinctions matter for inference.

The same algebra may support several logical matrices. One designated set may produce a weak consequence relation, while another produces a stronger one. A value that is inferentially invisible under one policy may become decisive under another.

The designated-set construction also permits paraconsistent, paracomplete, and many-valued behavior without naming those systems in advance. If several values are designated, formulas may remain accepted in situations that would be treated as inconsistent under a binary interpretation. If some valuations fail to designate either a formula or its anonymous complement-like transform, excluded-middle-like behavior may fail.

The experiment therefore treats designatedness as an observation policy. It determines which semantic differences survive into logic.

Two values may be distinct algebraically but indistinguishable with respect to all generated formulas and designatedness. If so, the matrix may contain redundant semantic states. This motivates the reducedness test discussed below.

The crucial methodological point is that logical status is not assigned directly to formulas. It emerges from the interaction among:

the operation table;

the designated subset;

the generated formula universe;

the valuation space.

A formula is valid only because every valuation places it in D. A consequence holds only because preservation of D is unavoidable across all valuations.

4.4. Emergence of semantic consequence

Given a matrix M and formulas $\Gamma \cup \{A\}$, consequence is computed exhaustively.

For every valuation v of the variables into the carrier, the program evaluates each formula. If all premises in Γ receive designated values and A also receives a designated value, that valuation supports the consequence. If all premises are designated but A is not, the valuation is a countermodel.

Thus $\Gamma \models A$ if and only if no countermodel exists.

For bounded formula universes, the computation is exact. There is no statistical estimation.

The initial consequence fingerprint may include:

zero-premise consequences, corresponding to valid formulas;

one-premise consequences, corresponding to direct inferential dependencies;

two-premise consequences, corresponding to genuinely joint support.

Higher-premise consequence can be added, but zero-, one-, and two-premise relations already reveal substantial structure.

A logical fingerprint can therefore be represented as a finite set:

$L(M) = \text{Valid}(M) \cup \text{Entail}_1(M) \cup \text{Entail}_2(M)$.

Matrices with identical fingerprints are observationally indistinguishable within the bounded universe.

This fingerprint is richer than an equation list. It records not only which formulas behave alike but how assumptions support conclusions.

From the consequence relation, mutual entailment is defined by:

$A \equiv B$

when $A \models B$ and $B \models A$.

This relation partitions formulas into inferential equivalence classes. The quotient formula structure is the first emergent logical object.

Directionality remains between the quotient classes. If [A] entails [B], an ordered relation exists from one inferential class to another. The quotient therefore resembles a finite consequence order or preorder after equivalent formulas are identified.

Validity classes occupy distinguished positions in this order. Premise combinations generate further closure. This provides the raw material from which proof rules can be extracted.

4.5. Cross-order recurrence of logical fingerprints

As in Algebraic Abiogenesis, one finite matrix is not enough. Small matrices can support accidental consequences. The central filter is recurrence across carrier sizes.

The initial exhaustive experiment considered all nondegenerate binary operations on two- and three-element carriers, paired with every nonempty proper designated subset.

For each matrix, consequence fingerprints were computed over a common formula universe. Distinct two-element theories were then compared with three-element theories.

The first scan produced exact cross-order matches. Every distinct two-valued theory found a counterpart among some three-valued matrices. Exactness here means that the complete bounded consequence fingerprint matched, not merely that the matrices shared a high percentage of rules.

This result was stronger than threshold-based clustering, but it raised an immediate concern. A three-valued matrix may simulate a two-valued one by splitting one semantic value into two indistinguishable copies. Such a recurrence is formally cross-order but not structurally new.

Consequently, raw recurrence cannot by itself certify logical persistence. The higher-order model must be shown to use its larger carrier irreducibly.

Cross-order analysis therefore proceeds in stages:

identify exact or high-stability fingerprint matches;

reject carrier relabelings;

test whether the larger matrix has indistinguishable values;

enlarge the formula universe;

increase the number of variables;

test substitution invariance;

seek separating countermodels.

Only recurrences surviving these stages count as robust.

4.6. Rejection of duplicated-value expansions

A three-valued matrix may be nothing more than a disguised two-valued matrix with one value duplicated.

Suppose values a and b behave identically under every operation and are either both designated or both undesignated. Collapsing them yields a smaller matrix with the same logical behavior. Treating such a matrix as independent cross-order evidence would exaggerate recurrence.

The appropriate test is logical reducedness.

An equivalence relation θ on the carrier is a congruence if it is preserved by every operation. It is compatible with D if each equivalence class lies entirely inside D or entirely outside D.

A matrix is reduced when the identity relation is the only congruence compatible with D.

If a nontrivial compatible congruence exists, the matrix can be quotiented to a smaller logical matrix without changing its consequence relation. Its extra semantic values are not logically distinguishable.

The exact reducedness test was applied to every three-valued candidate. This eliminated thousands of matrices whose apparent recurrence arose from value duplication or another reducible expansion.

The effect was substantial. Raw cross-order recurrence was widespread, but only a smaller subset survived reduction.

This illustrates the general necessity of adversarial filtering. A strong-looking numerical result may disappear once the most obvious trivial explanation is tested. The surviving result is smaller but more meaningful.

Reducedness also clarifies logical objecthood. A reduced matrix contains no semantic distinctions that the logic itself cannot detect. Every remaining value participates essentially in the designated-preservation structure.

4.7. Reduced logical matrices and robust theory clusters

After nonreduced expansions were removed, several exact cross-order recurrences remained.

These survivors were then tested against two independent enlargements.

The first expansion increased the formula universe while retaining two variables. This allowed deeper compositions and tested whether the initial fingerprint agreement was merely a consequence of shallow syntax.

The second expansion introduced a third variable. This tested whether recurrence depended on the restricted interaction patterns available with only two propositional variables.

Several candidate clusters failed under one or both enlargements. Three theory families survived both.

Each surviving family contained:

two-element representatives;

genuinely reduced three-element representatives;

exact agreement across the initial consequence universe;

continued agreement under deeper formulas;

continued agreement under the enlarged variable set.

The persistence of these clusters demonstrates that logical recurrence can survive both increased semantic order and increased syntactic complexity.

The result remains bounded. It does not prove that the consequence relations coincide for all formulas of arbitrary depth and any number of variables. It does, however, provide a much stronger certificate than a small truth-table coincidence.

The robust clusters can be represented by birth certificates containing:

the anonymous operation tables;

the designated subsets;

the formula generator;

the complete consequence fingerprints;

the reducedness proof;

the carrier-relabeling checks;

the enlarged-universe tests;

the surviving structural rules;

explicit countermodels for rejected candidates.

Such certificates make the discovery auditable and reproducible.

4.8. Post hoc recognition of conjunction-like and implication-like behavior

Only after the recurrent clusters were isolated were their structures interpreted.

One cluster exhibited consequences of the form:

$x \vDash b(x,y)$

$y \vDash b(x,y)$

$x, y \vDash b(x,y)$.

At the level of designatedness, the operation also behaved commutatively and idempotently.

These properties are retrospectively recognizable as join-like or disjunction-like behavior. Depending on the designated set, the same consequence pattern may also be represented by an AND table with the lower value designated rather than the upper value. This shows why operation names alone are insufficient. Logical behavior arises from the pair consisting of the operation and the designated policy.

A second cluster exhibited:

$\vDash b(x,x)$

$y \vDash b(x,y)$

$x, b(x,y) \vDash y$

$x, y \vDash b(x,y)$.

The third rule has a modus ponens-like form. The fourth resembles implication introduction under a two-premise semantic reading. The diagonal formula is universally designated.

This cluster is retrospectively implication-like.

A third cluster exhibited the argument-reversed counterpart:

$\vDash b(x,x)$

$x \vDash b(x,y)$

$y, b(x,y) \vDash x$

$x, y \vDash b(x,y)$.

This cluster behaves like a reversed implication.

The importance of these results is not the rediscovery of familiar connectives. Their value lies in showing that inferential roles can be recovered from anonymous finite semantics.

The operation was not selected because it looked like implication. It was selected because its consequence fingerprint recurred across reduced matrices and survived robustness tests. The implication interpretation came afterward.

This distinction separates calibration from target-driven confirmation.

4.9. Limits of bounded logical discovery

The logical experiments establish a computational bridge from anonymous operations to stable consequence relations, but several limitations remain.

First, the formula universes are finite. Agreement at increasing depths provides evidence of stability but not an automatic proof of unbounded identity.

Second, only finite matrices are examined. Some logics require infinite semantics, relational structures, topological models, or algebraic classes not captured by a single finite matrix.

Third, consequence has initially been computed over sets of premises. Set-based premises build weakening, contraction, and exchange into the representation. Substructural logics require multisets, sequences, resource annotations, or other premise structures so that these structural rules can themselves be discovered rather than assumed.

Fourth, the operation signature is initially small. A single anonymous binary operation can support rich behavior, but negation-like, modal, quantificational, and multi-connective systems require larger signatures or different semantic substrates.

Fifth, recurrence does not itself establish novelty. The surviving clusters may correspond to known finite matrix logics. Literature comparison must occur after discovery and must distinguish methodological novelty from the novelty of the resulting logic.

Sixth, proof theory requires more than semantic consequence. A logic becomes proof-theoretically mature only when compact schematic rules, soundness, completeness, normalization, cut behavior, and proof identity have been investigated.

These limitations define the next stage rather than undermining the current one.

Logical Abiogenesis has already demonstrated the following progression:

anonymous finite values
→ anonymous operations
→ designated subsets
→ formula evaluations
→ preservation-based consequence

→ mutual-entailment quotients
→ recurrent logical theories.

The next question is whether these semantic theories can generate their own proof calculi. That transition—from consequence to rules, proofs, and normalization—is the subject of the following section.

5. From Consequence to Proof Calculus

Logical consequence describes which conclusions are semantically forced by which premises. Proof theory asks whether those dependencies can be reproduced by a finite or recursively specified system of rules. The transition from consequence to proof calculus is therefore the point at which Logical Abiogenesis moves beyond semantic classification and begins to generate a formal system.

The central problem is not merely to list valid inferences. A bounded semantic universe may contain thousands of premise-conclusion relations. Most are redundant consequences of a much smaller set of structural and connective-specific rules. A proof calculus must compress this large relation into a compact generative basis.

Logical Abiogenesis approaches this task from below. It begins with a recurrent semantic consequence relation extracted from anonymous finite matrices. Formulas are quotiented by mutual consequence. Minimal entailment relations among the quotient classes are identified. Candidate introduction, elimination, and structural rules are inferred. Invalid generalizations are rejected by countermodels. The resulting calculus is then tested for soundness and completeness against the original semantic relation.

This process reverses the usual order of proof theory. Instead of supplying rules and proving that they match a semantics, it begins with anonymous semantic behavior and attempts to construct the rules that generate it.

5.1. Formula quotients under mutual consequence

Let F be a bounded universe of formulas generated from anonymous operation symbols. For formulas A and B, define

$A \equiv B$

when both

$A \models B$

and

$B \vDash A$.

This relation identifies formulas that occupy the same inferential position. They need not compute exactly the same value under every valuation. It is sufficient that each is designated whenever the other is designated in every relevant semantic context.

The quotient $F/\equiv$ replaces many syntactic formulas with a smaller set of inferential classes. These classes are the logical counterparts of equivalence classes in an algebraic quotient.

The quotient performs several important functions.

First, it removes syntactic redundancy. Different formulas that behave identically with respect to consequence are represented once.

Second, it exposes the order structure of inference. If $A \vDash B$, then the class [A] lies below or above [B], depending on the chosen orientation.

Third, it provides a finite state space on which proof-rule extraction becomes tractable.

Fourth, it distinguishes formula complexity from inferential complexity. A deeply nested formula may belong to the same class as a variable or a shallow term.

In the implication-like calibration, the two-variable formula universe collapsed into only six inferential classes. Increasing formula depth did not create new classes. The quotient stabilized.

This stabilization is significant. It indicates that the apparent growth of syntax was being absorbed by a fixed inferential structure. The semantic theory had discovered a finite logical object beneath an infinite or rapidly expanding formula grammar.

The quotient is therefore not merely a computational convenience. It is the first candidate proof-theoretic object generated by the system.

5.2. Extraction of axioms and inference rules

Once formulas have been quotiented, the consequence relation can be treated as a finite directed structure.

Zero-premise consequences identify axiom classes. A class [A] is axiomatic when

$\vDash A$.

One-premise consequences identify direct inferential dependencies. If

$A \vDash B$,

then a rule may permit passage from A to B.

Two-premise consequences identify genuinely joint inferences. If

$A, B \models C,$

but neither $A \models C$ nor $B \models C$, then the conclusion depends essentially on the combination of premises.

A naive calculus could include every valid consequence as a separate rule. Such a calculus would be complete by construction but would have no explanatory value. The goal is instead to identify a small set of primitive transitions whose closure reproduces the entire consequence relation.

The first compression step is transitive reduction. If A entails B and B entails C, then the direct relation from A to C need not be retained as a primitive rule. The Hasse-cover structure of the quotient removes consequences generated by chains of simpler entailments.

The second compression step removes premise redundancy. If $A, B \models C$ but $A \models C$, then B is unnecessary. The two-premise rule should not be treated as primitive.

The third step searches for recurrent rule shapes. Many instance-level consequences may share the same schematic structure. For example:

$A, b(A,B) \models B$

may occur across many substitutions for A and B. Those instances can be compressed into one metavariable rule.

A proof calculus emerges when a small family of such rules reproduces the complete bounded semantic closure.

5.3. Premise minimization and rule anti-unification

Premise minimization determines which assumptions are essential to an inference.

For each valid consequence

$\Gamma \models A,$

the system examines proper subsets of Γ. If some $\Gamma' \subset \Gamma$ also entails A, the larger premise set is not minimal. Only premise-minimal consequences are candidates for primitive rules.

This is especially important because monotonic consequence automatically generates many inflated inferences. If $A \models B$, then $A, C \models B$ for any additional premise C. Counting all such consequences would confuse structural monotonicity with logical content.

After minimal consequences are identified, the system seeks schematic generalizations.

Anti-unification is a method for finding the least general pattern shared by several expressions. Given multiple consequence instances, the algorithm replaces differing subterms with metavariables while preserving common structure.

For example, the instances

$x, b(x,y) \models y$

and

$b(x,z), b(b(x,z),y) \models y$

may suggest the schema

$A, b(A,B) \models B$.

The candidate schema is then tested across all substitutions available in the formula universe and across all matrices in the recurrent semantic cluster.

If it fails, a counterexample records the boundary of the attempted generalization. If it succeeds repeatedly under enlarged term universes, it becomes a stronger candidate rule.

Rule discovery therefore alternates between generalization and falsification:

collect instances;

anti-unify their forms;

test the generalized schema;

generate countermodels or counterinstances;

refine or reject the rule.

This process mirrors theory peeling in Algebraic Abiogenesis. A compact rule explains many observed inferences, its consequences are removed, and the residual consequence relation is examined for additional structure.

5.4. Introduction and elimination behavior

Connective meaning is often determined proof-theoretically by introduction and elimination rules.

An introduction rule specifies the conditions under which a compound formula may be inferred. An elimination rule specifies what may be derived from that compound formula.

In the conjunction-like cluster, the anonymous operation b supports behavior of the form

$A, B \models b(A,B)$,

together with

$b(A,B) \models A$

and

$b(A,B) \models B$.

Only after these rules are discovered is b recognizable as conjunction-like.

The operation is not defined by a truth table alone. Its inferential role is determined by the relation between the compound formula and its components.

In the implication-like cluster, the rule

$A, b(A,B) \models B$

functions as an elimination rule.

A corresponding introduction pattern appears as

$\Gamma, A \models B$

if and only if

$\Gamma \models b(A,B)$.

This relation connects an operation inside the formula language with movement of assumptions in the consequence relation. It is stronger than a collection of truth-table identities. It gives the operation a structural role in proofs.

The discovery of paired introduction and elimination behavior is a major milestone. It shows that anonymous semantic operations can generate rule-governed meaning.

A mature Logical Abiogenesis system should search for such adjunctions automatically. It should examine whether adding a premise corresponds to applying an anonymous operation in the conclusion, whether a compound formula can be decomposed into its parts, and whether these transformations are stable across models and substitutions.

5.5. Modus ponens and implication adjunction

The implication-like cluster yielded a particularly important result.

The system tested the equivalence

$\Gamma \cup \{A\} \models B$

if and only if

$\Gamma \models b(A,B)$.

This is a semantic deduction property. It means that the anonymous binary operation internalizes the addition of a premise.

The property was tested across enlarged formula universes and recurrent matrices. No failures were observed in the calibration.

The same cluster also supported

$A, b(A,B) \models B$.

This is the semantic form of modus ponens.

Together, these rules define a strong implication-like role:

premise discharge corresponds to forming b(A,B);

application of b(A,B) together with A yields B.

The diagonal formula

b(A,A)

was also valid throughout the tested universe.

These relations were not inserted as targets. They were identified after the recurrent consequence theory had already been isolated.

This matters because implication is not merely a Boolean function. Its deeper proof-theoretic significance lies in the way it relates assumptions, conclusions, and derivability.

The implication-like operation discovered by Logical Abiogenesis therefore has both semantic and proof-theoretic coherence.

A similar analysis can be applied to other recurrent operations. One may search for conjunction-like pairing, disjunction-like case behavior, negation-like incompatibility, equivalence-like bidirectionality, or modal-like closure.

The goal is not to force every operation into a familiar role. It is to detect which structural relationships, if any, the operation sustains.

5.6. Countermodels for invalid generalizations

A candidate rule must be challenged, not merely illustrated.

Suppose a pattern appears valid across several formulas:

A, b(A,B) ⊨ B.

A weak test would confirm it on the observed examples. A stronger test searches all formulas and valuations in the current universe. A still stronger test searches additional finite matrices satisfying the previously discovered theory.

If the rule fails, the system returns a countermodel containing:

the finite carrier;

the anonymous operation table;

the designated subset;

the substitution instance;

the valuation;

the designated premises;

the undesignated conclusion.

This certificate shows exactly why the generalization is invalid.

Countermodels are also used to test premise deletion. If a candidate two-premise rule appears valid, the system checks whether either premise can be removed. A countermodel to the reduced rule proves that both premises are necessary.

They test rule reversal as well. From A ⊨ B it does not follow that B ⊨ A. A countermodel prevents the system from turning an order relation into an equivalence.

Adversarial model generation is especially important when the initial semantic cluster is small. A rule may hold in every sampled matrix but fail in another model of the same apparent basis theory.

Countermodel-guided refinement therefore separates three classes of rule:

empirically valid in the original sample;

valid throughout the enlarged finite model class;

derivable from a proposed basis.

Only the latter two support strong proof-theoretic claims.

5.7. Bounded soundness and completeness

A generated calculus must satisfy two distinct requirements.

Soundness requires that every derivable consequence be semantically valid.

If

$\Gamma \vdash A$,

then

$\Gamma \models A$.

Completeness requires that every semantic consequence be derivable.

If

$\Gamma \models A$,

then

$\Gamma \vdash A$.

For a finite quotient universe, both conditions can be tested exhaustively.

The proof engine begins with the generated axiom and rule set. For every premise context, it computes the fixed point of all applicable rules. The resulting derivable conclusions are compared with the complete semantic consequence table.

A soundness failure occurs when the proof engine derives a class not semantically entailed by the premises.

A completeness failure occurs when the semantic relation contains a consequence absent from the proof closure.

In the bounded Logical Abiogenesis experiments, the generated rules reproduced every zero-, one-, and two-premise semantic consequence over the tested quotient.

This does not automatically prove completeness for arbitrary formulas. It does establish that the rule basis is not merely suggestive. Within the certified universe, it generates exactly the target consequence relation.

The bounded certificate should include:

the quotient classes;

the representative formulas;

the semantic consequence table;

the rule basis;

the complete proof closure;

all soundness comparisons;

all completeness comparisons.

Such a certificate is the logical counterpart of an algebraic normal-form audit or an exact circuit-minimization witness.

5.8. Rank-two closure and finite proof-theoretic certification

The implication-like calibration produced a stronger finite result than ordinary bounded testing.

With two propositional variables, all generated formulas collapsed into six semantic classes. Increasing formula depth did not create new classes. The six classes were also closed under the anonymous binary operation.

This closure permitted the entire rank-two logical algebra to be treated as a finite object, even though syntactic formula generation could continue indefinitely.

All possible premise contexts over the six classes were enumerated. Since there are six classes, there are $2^6 = 64$ set-valued contexts.

The proof system used only three general principles:

assumption;

implication elimination;

implication introduction.

For each of the 64 contexts, the syntactic closure was computed and compared with semantic consequence.

There were no missing semantic consequences and no unsound derivations.

This provides an exact completeness certificate for the entire two-generator quotient, not merely for formulas up to one chosen depth.

The result remains rank-specific. It does not prove that the same finite quotient or completeness relation holds for three or more generators. Nevertheless, it shows how finite closure can convert an apparently bounded experiment into an exact theorem about an entire generated subtheory.

This technique should be generalized. Whenever quotient-class growth stabilizes and the operation closes on the resulting classes, the system can attempt exhaustive proof-theoretic certification.

Such finite closures may act as laboratories for discovering candidate general theorems.

5.9. The path toward unbounded completeness and normalization

The next challenge is to move from finite certification to general proof theory.

A full result would require soundness and completeness for arbitrary formulas, arbitrary finite premise sets, and unbounded variable rank.

Several routes are available.

One route is a canonical model construction. Formulas are quotiented by provable equivalence, and the quotient is used as a semantic algebra. If every nonderivable consequence fails in this canonical model, completeness follows.

A second route uses maximal consistent or deductively closed sets. A failed derivation is extended to a theory separating the premises from the conclusion, which then induces a countermodel.

A third route relies on algebraic completeness. If the inferred rules correspond to an equational or order-theoretic class, the proof system may be shown complete with respect to the associated algebraic semantics.

A fourth route exploits the finite-model property. If every invalid consequence has a finite countermodel of bounded or recursively searchable size, finite model generation can support completeness.

Normalization is a further goal. Introduction followed immediately by elimination may create proof detours. A normalization procedure should remove such detours without changing the conclusion.

If all proofs can be reduced to canonical forms, the system gains:

proof identity criteria;

decidability of derivability in suitable fragments;

subformula properties;

proof compression;

structural explanations of inference.

Cut elimination is especially important. A cut rule allows an intermediate lemma to connect two derivations. Cut admissibility shows that such intermediate formulas are not fundamentally required for provability.

Logical Abiogenesis should not assume cut elimination. It should search for transformations that remove intermediate formulas and test whether derivability is preserved.

The larger developmental sequence is:

anonymous semantic matrices
→ recurrent consequence
→ formula quotients
→ schematic rules
→ proof closure
→ soundness and completeness
→ proof normalization
→ canonical proofs.

At that point, the process would have generated not merely a logical theory but a mature proof system from finite operational observations.

The central result of the present stage is already substantial: inferential rules can be extracted from anonymous semantics and validated against the consequence relation that gave rise to them. The rules are not external descriptions imposed after the fact. They are compressed generators of the discovered logical behavior.

6. Circuit Abiogenesis from NAND Composition

Circuit Abiogenesis applies the same operation-first methodology to computation. Instead of beginning with a named Boolean function and asking how to implement it, the process begins with primitive signals, one universal gate, recursive composition, and behavioral equivalence. Circuits are generated before their purpose is known. Their truth functions, reusable motifs, multi-output relationships, and minimal realizations are discovered afterward.

The foundational experiment uses NAND because a single NAND operation is functionally complete. Any Boolean function can be expressed using only NAND gates. This makes NAND an unusually clean primitive substrate: the generator is simple, but the closure of its compositions contains the full finite Boolean function space.

The novelty of the experiment does not lie in the fact that NAND can construct familiar gates. That result is classical. The methodological importance lies in reversing the normal design

process. The system is not instructed to construct AND, OR, XOR, parity, a multiplexer, a half-adder, or a full-adder. It exhaustively or systematically generates circuits, quotients them by identical truth behavior, determines exact minimal implementations, and searches for recurrent internal motifs and multi-output conservation relations. Familiar components appear only as retrospective interpretations of previously anonymous computational objects.

Circuit Abiogenesis therefore provides a bridge between algebraic theory formation and neural computation. A circuit is a generated expression with reuse. Behavioral equivalence replaces syntactic identity. Minimal netlists become canonical representatives. Recurrent subgraphs become modules. Multi-output compression reveals when several functions belong to one integrated computational object.

6.1. NAND as a primitive generator rather than a named design language

The NAND operation on Boolean inputs is defined by the truth condition that its output is false only when both inputs are true. In conventional digital logic, NAND is introduced as a known gate with a known symbol, known properties, and known implementation patterns.

Circuit Abiogenesis uses the same operation table but suppresses the usual design vocabulary during generation. NAND is treated as a primitive binary transformation on truth functions. Its compositions are not initially described as negations, conjunctions, disjunctions, exclusive disjunctions, selectors, or arithmetic circuits.

Let $x_1, \ldots, x_n$ denote the primary input signals. Each signal is represented by its complete truth vector over all 2^n input valuations. A NAND gate combines two available signals f and g to form a new signal:

$N(f,g) = \neg(f \wedge g)$.

At the implementation level, the operation is familiar. At the discovery level, however, the system records only the resulting truth vector and the graph that produced it.

This distinction matters because a target-driven circuit synthesizer begins with a desired truth table. It asks for the least costly circuit computing that function. Circuit Abiogenesis instead asks which functions appear, which ones share intermediate structure, which motifs recur, and which combinations form higher-order modules.

NAND is therefore a primitive generator, not a catalogue of intended constructions. It is analogous to the anonymous operation table in Algebraic Abiogenesis. The closure of the operation generates a computational ecology from which stable objects are extracted.

6.2. Recursive circuit construction and truth-vector equivalence

The initial computational universe contains only the primary input functions. New functions are formed recursively by applying NAND to any two previously available signals.

If C_o is the set of primary inputs, then a simple recursive construction has the form:

$C_{k+1} = C_k \cup \{N(f,g) : f,g \in C_k\}$.

This recursion generates both functions and implementations. Many different gate graphs can compute the same truth function. The raw graph universe therefore grows much faster than the semantic function universe.

For n Boolean inputs, only

$2^{(2^n)}$

distinct Boolean functions exist. For two inputs there are 16 functions. For three inputs there are 256. The number of syntactically distinct NAND circuits is unbounded.

Truth-vector equivalence provides the quotient:

$C_1 \sim C_2$

when both circuits return the same output on every input valuation.

This equivalence is exact. No approximation or sampling is required for small input arities.

The quotient maps an enormous circuit space onto a finite function space. Each equivalence class contains every NAND realization of one Boolean function. Minimal circuit synthesis then asks for the least costly representative of each class.

This structure directly parallels Algebraic Abiogenesis:

formal terms correspond to circuit graphs;

term-function equality corresponds to truth-vector equality;

equivalence classes correspond to Boolean functions;

normal forms correspond to minimal or canonical netlists.

The principal difference is reuse. Algebraic terms are usually trees, whereas circuits can be directed acyclic graphs in which one intermediate result feeds several later gates. This difference becomes central to modularity and compression.

6.3. Formula trees versus shared directed acyclic graphs

A formula tree duplicates every repeated subexpression. If the same intermediate computation is needed twice, it appears twice as separate branches. A circuit directed acyclic graph permits one computed signal to be reused.

Suppose a four-gate subexpression computes an intermediate function h. A formula-tree implementation requiring h in two locations pays the four-gate cost twice. A DAG implementation computes h once and fans its output into both locations.

The distinction between trees and DAGs is therefore the distinction between expression complexity and circuit complexity.

For a formula tree, cost may be measured by the number of NAND occurrences. For a DAG, cost is the number of distinct gates.

The reduction produced by sharing is not merely an engineering optimization. It is evidence that a subcomputation has become a reusable object. A repeated term becomes a module when one canonical realization can support multiple downstream uses.

In the three-input experiments, shared DAGs reduced the costs of many functions relative to their minimal formula trees. Some functions showed little or no gain because their minimal formulas contained little reusable structure. Others showed substantial reduction.

Three-input odd parity provides a strong example. Its minimal NAND formula tree requires considerably more NAND occurrences than its minimal shared DAG. The reduction reveals repeated structure that becomes visible only when the representation permits reuse.

This suggests a general principle:

modularity emerges when semantic compression and graph reuse align.

A recurring subfunction is not automatically a module. It becomes one when its reuse lowers total implementation cost across several contexts or outputs.

6.4. Exact discovery of two-input Boolean functions

The two-input case provides the smallest complete calibration.

There are 16 Boolean functions of two variables. Starting only from the input functions x and y and recursively composing NAND gates, the search reaches all 16.

The functions can be grouped retrospectively by their minimum NAND-gate counts. Primary inputs require no gates. Several simple functions appear after one gate. Additional functions appear at progressively larger costs, with the last function reached at five gates.

Only after the complete ecology is generated are familiar names assigned. Among the discovered functions are those later recognized as:

negation of an input;

NAND;

AND;

OR;

implication;

exclusive disjunction;

equivalence;

constants.

The discovery process does not treat these names as objectives. Each function is initially identified only by its truth vector.

Exact breadth-first exploration supplies a minimum-gate certificate. If a function first appears at cost k after all states of lower cost have been exhausted, then no smaller NAND circuit exists under the chosen circuit model.

The two-input calibration establishes several capabilities:

semantic quotienting of circuits;

complete functional coverage;

exact minimum-cost representatives;

retrospective gate recognition;

comparison between separate and joint output synthesis.

Although the results are known in classical logic design, the experiment validates the abiogenetic pipeline in a domain where exhaustive certification is possible.

6.5. Exact discovery of three-input Boolean functions

The three-input case is substantially richer. There are 256 Boolean functions, and the space of candidate circuits grows rapidly.

Exact search nevertheless recovered a minimum shared-DAG cost for every function. The resulting cost distribution is not uniform. Many functions are reachable with relatively few gates. A smaller number require deeper and more highly organized circuitry.

The unique hardest function in the exact search is the detector that outputs one exactly when one and only one of the three inputs is high. Its minimum implementation requires ten NAND gates under the specified model.

This result provides more than a witness circuit. Exhaustive lower-cost rejection proves that no circuit using nine or fewer NAND gates computes the same truth function.

Other complex functions appear at costs below ten. Odd parity, majority, and related symmetric functions occupy the higher end of the distribution but differ in their exact complexity and opportunities for shared structure.

The complete 256-function ecology creates a finite landscape of computational difficulty. Every function has:

a truth-vector identity;

a minimum NAND cost;

at least one exact witness netlist;

a family of equivalent alternative implementations;

relationships to simpler internal functions.

This landscape can be studied independently of familiar logical names. Symmetry, algebraic degree, support structure, sensitivity, and circuit complexity can be compared after the search.

The important methodological result is that computational objects can be exhaustively born from recursive gate composition and then organized by exact semantic and structural properties.

6.6. Minimal circuit complexity and certified lower bounds

A circuit witness proves only that a function can be computed with a given number of gates. It does not prove minimality.

Exact minimality requires eliminating all smaller possibilities. Breadth-first state-space search provides one route. The search enumerates circuit states in increasing gate count, while semantic deduplication prevents repeated exploration of equivalent signal sets.

A circuit state can be represented by the collection of truth functions currently available from the primary inputs and constructed gates. Adding one NAND gate augments the state with one new function.

If a target function first appears in a state at depth k, and every state of smaller depth has been exhaustively processed, then k is an exact lower and upper bound.

For multi-output circuits, the target is not one truth function but a set of required outputs. The search asks for the smallest state containing all of them. Shared intermediates can reduce the joint cost below the sum of separate minimum costs.

Exact lower bounds are essential because heuristic synthesis can produce compact circuits without revealing whether further reduction is possible. For abiogenetic analysis, exactness distinguishes a canonical minimal representative from one merely convenient implementation.

The lower-bound certificate should include:

the search model;

the state representation;

the gate-count ordering;

the symmetry reductions;

the semantic deduplication rule;

the first state exposing the function or output set;

confirmation that all lower-depth states were exhausted.

Such certificates make circuit complexity results reproducible rather than impressionistic.

6.7. Recurrent subcircuits and modular reuse

Once minimal circuits are available, the next question is whether certain internal structures recur across many functions or within larger circuits.

A recurrent subcircuit is a graph motif that repeatedly computes a useful intermediate function. Frequency alone is not sufficient. A motif may be common merely because of search bias or syntactic convention. Stronger evidence of modularity includes:

appearance in independently minimized circuits;

reuse within one multi-output circuit;

replacement by an equivalent compact component;

reduction in total gate count;

stability under input permutation;

recurrence across larger arities.

The full-adder circuit revealed a particularly clear motif: a four-NAND construction appears twice, first combining two primary inputs and then combining the resulting signal with the third input.

Retrospectively, the four-gate motif computes exclusive disjunction. Within the discovery process, however, it first appears as a repeated truth-function-producing subgraph.

The repeated use is important. The motif is not merely one function among 256. It has become a compositional building block inside a larger arithmetic module.

A module may therefore be defined operationally as a subcircuit whose abstraction reduces description length or implementation cost across several contexts.

This definition aligns circuit formation with neural object formation. A learned neural feature or circuit becomes object-like when its output is reused across tasks or downstream computations.

6.8. Emergence of XOR-like motifs

Exclusive disjunction occupies a special position in arithmetic circuits because it represents addition modulo two.

In the NAND ecology, XOR is not primitive. It emerges as a four-gate construction.

A common four-gate motif has the form:

t = NAND(a,b)

u = NAND(t,a)

v = NAND(t,b)

out = NAND(u,v).

The output is one exactly when a and b differ.

The significance of this motif is not the rediscovery of a standard NAND identity. Its significance is its repeated structural role.

In the full-adder circuit, one copy computes the parity of two inputs. A second copy combines that parity with the third input. This recursive reuse generates the three-input sum bit.

The motif therefore displays three forms of stability:

semantic stability, because it computes the same truth function under repeated instantiation;

structural stability, because the same graph pattern recurs;

functional stability, because its output supports a larger module.

This is a simple example of hierarchical object formation. NAND gates generate a recurrent four-gate motif. The motif becomes a component. Two instances of the component participate in a nine-gate full-adder.

At higher scales, the full-adder itself can become a component in a multi-bit arithmetic circuit.

The hierarchy is:

gate;

motif;

module;

system.

Circuit Abiogenesis seeks to detect these transitions automatically.

6.9. Multi-output conservation and arithmetic modules

Single-output function discovery identifies truth functions. Multi-output analysis can reveal integrated systems whose outputs jointly encode a conserved quantity or transformation.

For two Boolean inputs a and b, the number of high inputs is 0, 1, or 2. A two-output circuit can encode this quantity in binary using one low-order output and one high-order output.

The relation is:

$a + b = s + 2c$,

where s and c are Boolean output values interpreted numerically.

An exhaustive scan over output pairs identifies exactly one pair satisfying this conservation law for every input valuation. Retrospectively, s is XOR and c is AND. Together they form a half-adder.

For three inputs a, b, and c_{in}, the corresponding relation is:

$a + b + c_{in} = s + 2c_{out}$.

Again, exactly one output pair satisfies the relation across all valuations. Retrospectively, s is odd parity and c_{out} is majority. Together they form a full-adder.

The arithmetic interpretation is assigned after the output relation is found. The circuit generator itself does not begin with a half-adder or full-adder target.

The conservation criterion is nevertheless an observation policy. It asks which output combinations preserve an input quantity. It therefore guides module recognition without prescribing the internal circuit structure.

The method can be generalized. Multi-output modules may be discovered through:

conservation of weighted counts;

bijective input-output transformations;

routing relations;

state-transition consistency;

error-correcting redundancy;

shared compression advantage;

recurrent composition into larger networks.

Arithmetic modules are one class within a larger ecology of coordinated output systems.

6.10. Post hoc recognition of half-adders, full-adders, and multiplexers

The discovered half-adder requires fewer gates when synthesized jointly than when its two outputs are synthesized separately. The shared circuit uses five NAND gates rather than the sum of four gates for XOR and two for AND.

The full-adder shows a larger advantage. Its outputs require eight and six gates when minimized independently, but only nine gates jointly. Shared intermediate functions save five gates.

This difference is evidence that the output functions belong naturally to one module. They are not merely two unrelated functions placed beside each other. Their joint implementation contains substantial common structure.

Multiplexer-like functions also emerge from the three-input ecology. A selector input determines which of the other two inputs is transmitted to the output. Under input permutations and selector orientation, several routing variants appear.

Each can be implemented with four NAND gates.

Again, the familiar name is retrospective. The initial discovery criterion identifies functions whose restrictions under two selector values reproduce two other input signals.

The emergence of adders and multiplexers demonstrates two different forms of computational organization:

arithmetic aggregation;

conditional routing.

These are foundational operations in digital systems. Their appearance from one primitive gate suggests that a richer module ecology could recover larger classes of computation without beginning from a conventional hardware vocabulary.

The methodological caution remains important. NAND adders and multiplexers are known. Their rediscovery validates the machinery but does not establish novelty in circuit design.

Novelty would require one or more of the following:

a new minimum circuit;

an unfamiliar multi-output module with strong reuse;

a new composition principle;

a previously unknown cost relation;

a target-free discovery that outperforms established synthesis methods;

a module whose usefulness becomes evident only in larger generated systems.

6.11. Toward autonomous circuit invention

The next stage is to move from exhaustive small-function calibration to open-ended circuit development.

The first extension is hierarchical abstraction. Once a recurrent subcircuit is identified, it should be promoted to a new component symbol. Future search can operate over both primitive NAND gates and discovered modules.

The second extension is multi-level compression. A new component is justified when replacing repeated subgraphs with one abstract symbol reduces total description length across the circuit ecology.

The third extension is multi-output birth. The system should search not only for individual functions but for sets of outputs whose joint implementation is unusually efficient or whose behavior satisfies a stable relation.

The fourth extension is recursive module composition. Half-adders and full-adders can be connected into larger arithmetic structures. The system should determine which composition patterns minimize gates, depth, fan-out, delay, or wiring.

The fifth extension is environmental selection. Circuits can be evaluated under task distributions rather than one fixed target. Modules that recur across many tasks receive greater weight.

The sixth extension is fault and perturbation analysis. Stable circuits may preserve behavior under gate failure, noisy inputs, or restricted resources. Robustness can become part of object selection.

The seventh extension is architecture-independent equivalence. A transistor network, NAND netlist, neural microcircuit, and symbolic formula may implement the same function. Functional quotienting can identify their shared computational object.

The long-term goal is not merely automated synthesis from specifications. It is autonomous circuit ecology:

primitive gates generate functions;

functions generate motifs;

motifs become modules;

modules compose into machines;

machines are selected for conservation, compression, robustness, and reuse;

new computational objects are recognized only after their emergence.

This program connects directly to Neural Abiogenesis. Neural networks may contain distributed analogues of shared subcircuits. Training may discover recurrent internal computations because they reduce loss across many contexts. Compression may then consolidate those computations into compact modules.

NAND Circuit Abiogenesis offers an exact finite laboratory for studying this process. Every truth function can be observed, every equivalence can be certified, every small circuit can be minimized, and every reuse advantage can be measured.

The broader hypothesis is that intelligence may construct internal computation by the same sequence:

recursive composition;

behavioral equivalence;

compression;

module formation;

hierarchical reuse.

Circuit Abiogenesis therefore serves not only as an extension of Algebraic Abiogenesis but as a controlled model of how complex computational structure can arise from a minimal operational substrate.

7. Recursive Abiogenesis: The Minimal Theory of Iterated Systems

Recursion introduces a new mode of generative closure. Algebraic terms grow by composing operations into expressions. Circuits grow by composing gates into directed graphs. Recursive systems instead generate structure through repeated application of an update rule to a changing state. The output of one step becomes the input of the next:

$x_{n+1} = f(x_n)$.

This simple relation can generate fixed points, cycles, transient histories, attractor basins, symbolic sequences, bifurcations, and chaotic behavior. None of these structures needs to be specified in advance. They arise from the repeated action of the update rule and from the observational policy used to compare trajectories.

Recursive Abiogenesis asks what can be discovered when the update map is treated anonymously. Rather than selecting a known recurrence and studying its consequences, the method begins with a space of possible maps, a set of initial states, iteration, and exact or approximate comparison of orbit behavior. Stable dynamical objects are then formed by identifying states, trajectories, or parameter values that become behaviorally interchangeable.

The finite case provides the cleanest foundation. Every unary map on a finite carrier can be enumerated, every orbit can be computed exactly, and every recurrent structure can be certified. This creates a minimal theory of recursion before geometry, continuity, probability, or physical interpretation is introduced. The same architecture can later be transferred to the real line, the complex plane, cellular automata, and neural optimization.

7.1. Recursion as a distinct generative closure

Recursion differs from ordinary algebraic composition because it generates a temporal lineage. In a term algebra, an expression records how operations were composed. In a recursive system, the state itself changes through repeated application:

$x_0, x_1, x_2, x_3, \ldots$

where

$x_n = f^n(x_o)$.

The generative object is therefore not only a formal expression but an orbit.

The update rule may be deterministic or stochastic, single-valued or multivalued, autonomous or parameter-dependent. The minimal deterministic case uses one state space X and one map

$f: X \to X$.

Even this elementary system supports several layers of structure.

A state may return to itself after one step. A group of states may repeat periodically. A state may wander for several steps before entering a recurrent cycle. Several starting states may merge into one future history. Different cycles may attract disjoint regions of the carrier. The same map may therefore partition its state space into dynamically distinct components.

The recursion operator also creates a natural hierarchy of observations. One may compare states by their immediate images, by their complete future trajectories, by their eventual cycles, or by coarser statistics such as period length and transient duration.

Each observation policy produces a different quotient.

Recursion is thus not merely repeated calculation. It is a mechanism for constructing identity through future behavior.

7.2. Anonymous unary maps on finite carriers

Let X be a finite carrier of size n. A unary map is any function

$f: X \to X$.

There are n^n such maps.

For small orders, the complete map space is easily enumerated:

order 2 contains 4 maps;

order 3 contains 27 maps;

order 4 contains 256 maps;

order 5 contains 3,125 maps;

order 6 contains 46,656 maps.

Each map can be represented as a directed graph in which every state has exactly one outgoing edge. The map is initially anonymous. It is not labeled constant, periodic, contracting, mixing, or reversible.

The recursive experiment proceeds by iterating every state under every map. Because the carrier is finite, each orbit must eventually repeat. If an orbit revisits a previous state, all subsequent states repeat with a fixed period.

This gives every starting state a finite certificate containing:

its transient path;

the first repeated state;

its eventual cycle;

the cycle period;

the transient length;

the basin to which it belongs.

Carrier relabeling must be removed from the analysis. Two maps related by a permutation of X have the same dynamical structure. Canonicalization under relabeling converts many labeled tables into one functional graph type.

The resulting ecology can be organized by invariants such as:

number of fixed points;

number of cycles;

cycle-length multiset;

basin-size multiset;

maximum transient depth;

branching structure of predecessor trees;

reversibility;

image size.

No dynamical category needs to be named beforehand. The categories arise from exact orbit structure.

7.3. Orbits, transient states, and periodic cycles

For a starting state x, the orbit is

$Orb(x) = \{x, f(x), f^2(x), \ldots\}$.

Since X is finite, there exist integers $\mu \geq 0$ and $\lambda \geq 1$ such that

$f^{(\mu+\lambda)}(x) = f^{\mu}(x)$.

The integer μ is the transient length. The integer λ is the eventual period.

If $\mu = 0$, the starting state already lies on a cycle. If $\lambda = 1$, the cycle is a fixed point.

The pair (μ, λ) provides a coarse orbit signature. It distinguishes immediate recurrence from long transients and short cycles from long ones.

However, two states may share the same pair while entering different cycles. A stronger signature records the canonical identity of the eventual cycle. A still stronger signature records the entire rooted predecessor structure feeding that cycle.

This hierarchy illustrates how observation controls object formation.

Under a weak policy, all states with the same period may be equivalent.

Under a stronger policy, only states entering the same cycle are equivalent.

Under complete future equality, two states are equivalent only when their trajectories eventually merge exactly.

Transient states are dynamically asymmetric. They influence future behavior but are never revisited after departure. Recurrent states belong to cycles and are visited indefinitely.

This distinction emerges automatically from iteration. No prior concept of stability is required.

The decomposition of a finite functional graph into cycles with rooted in-trees is therefore the first structural theorem of Recursive Abiogenesis.

7.4. Attractors and basins as emergent objects

In a deterministic finite system, every cycle is an attractor in the weak sense that some set of states eventually enters it.

The basin of a cycle C is

$B(C) = \{x \in X : f^k(x) \in C \text{ for some } k \geq 0\}$.

Basins partition the carrier. Every state belongs to exactly one basin.

This partition creates a quotient:

$x \sim y$

when x and y eventually enter the same cycle.

The quotient classes are not defined by present-state similarity. Two states may be far apart under any external labeling yet belong to the same basin because their futures converge to the same recurrent structure.

The attractor cycle becomes the canonical representative of its basin. Transient histories are compressed into their eventual recurrent destiny.

A finer quotient can preserve phase. Two states are equivalent only when their trajectories eventually coincide at the same point in the cycle.

A coarser quotient may ignore cycle identity and preserve only period length.

These alternatives demonstrate that even in the simplest recursive system, several legitimate ontologies are available. Objecthood depends on what aspects of the future are retained.

The basin quotient is especially important because it generalizes naturally. In continuous systems, exact convergence may be replaced by asymptotic proximity. In neural training, many parameter states may belong to the same basin of functionally equivalent solutions. In cellular automata, many initial patterns may flow toward the same periodic or propagating regime.

Attractors are therefore recursive objects produced by repeated loss of historical distinction.

7.5. Eventual equivalence and orbit quotients

Several equivalence relations can be defined on recursive states.

The strongest is orbit equality:

$x \sim_0 y$

when

$f^k(x) = f^k(y)$

for every $k \geq 0$.

This usually identifies only identical states.

A more useful relation is eventual equality:

$x \sim_1 y$

when there exists N such that

$f^k(x) = f^k(y)$

for every $k \geq N$.

This identifies states whose trajectories eventually merge.

A still coarser relation is common-attractor equivalence:

$x \sim_2 y$

when x and y eventually enter the same cycle, possibly at different phases.

One may also define period equivalence:

$x \sim_3 y$

when their eventual cycles have the same length.

Each relation discards additional information. The quotient grows smaller as the observation becomes coarser.

These relations form a refinement hierarchy:

exact state identity;

complete orbit identity;

eventual trajectory identity;

common-attractor identity;

common-period identity.

This hierarchy is a dynamical version of the refinement lattices that appear in general object formation. An observation policy determines how much historical information is retained.

The quotients also support induced dynamics. Under eventual-equality quotienting, transiently distinct states may collapse into one future state. Under basin quotienting, every basin becomes fixed at the quotient level because the entire long-term behavior is represented by one attractor class.

Thus quotienting can simplify dynamics radically. A complicated state graph may become a small transition system among attractor types.

This is recursive compression in an exact finite form.

7.6. Recursive depth, stability, and structural recurrence

Finite recursion reaches eventual periodicity after a bounded number of steps, but discovery still depends on observational depth.

If the system is examined for too few iterations, long transients may be mistaken for nonrepeating complexity. Two states that eventually merge may appear distinct. A map with a long cycle may appear aperiodic.

The required observation depth is bounded by the carrier size. An orbit on an n-element carrier must repeat within at most n + 1 observations. Exhaustive finite analysis can therefore certify eventual structure exactly.

In larger or continuous systems, no such simple bound may be available. Stability must then be assessed through increasing time horizons.

A recursive structure is more credible when its classification persists as iteration depth increases. Candidate attractors, orbit classes, and recurrence statistics should be recomputed at multiple horizons.

Structural recurrence also refers to repetition across map orders or parameter families. A dynamical motif may appear in many unrelated maps:

a fixed point with a branching predecessor tree;

a two-cycle with asymmetric basin sizes;

nested transient chains;

multiple attractors separated by basin boundaries;

reversible cycles with no transients.

These motifs can be treated as dynamical species. Their recurrence across carrier sizes suggests general combinatorial structure rather than isolated examples.

Explanation subtraction can also be applied. If a map's behavior is fully explained by being a permutation, then its cycles require no additional theory of attraction. After permutation dynamics are separated, the residual ecology may reveal irreversible merging, basin asymmetry, or unusually long transient structure.

Recursive fertility can be measured by:

diversity of orbit types;

depth of transient structure;

number of attractors;

sensitivity to initial states;

compressibility of the transition graph;

reusability of discovered motifs;

persistence across carrier orders.

These criteria distinguish dynamically barren maps from maps supporting richer organization.

7.7. From finite iteration to continuous-state dynamics

The finite unary-map experiment provides a minimal exact foundation, but many important recursive systems use infinite or continuous state spaces.

On the real line or complex plane, a map

$f: X \to X$

may produce convergence, divergence, periodicity, quasiperiodicity, sensitive dependence, or chaotic attractors.

Exact state equality becomes rare. Observation must therefore become approximate or structural.

Possible orbit signatures include:

bounded versus escaping;

convergence to a fixed point;

estimated period;

Lyapunov behavior;

frequency of visits to regions;

symbolic itinerary;

invariant-measure statistics;

topological recurrence;

basin membership.

Each signature defines a policy-relative equivalence among trajectories.

The passage to continuous spaces introduces several complications.

Numerical precision can create false cycles. Finite iteration can misclassify long transients. Nearby initial conditions may diverge rapidly. Boundary points may require arbitrarily deep computation. Approximate equality may fail to be transitive.

These problems require interval arithmetic, symbolic methods, multiple precision, perturbation tests, and explicit uncertainty certificates.

Parameterized recursion adds another level. Let

$x_{n+1} = f_p(x_n)$,

where p belongs to a parameter space.

The parameter itself becomes a substrate. Two parameter values may be equivalent when they produce the same qualitative orbit structure. The parameter space can then be quotiented into dynamical regimes.

Boundaries between these regimes may become geometrically complex. Fractal parameter sets arise when arbitrarily small parameter changes alter long-term behavior.

The finite theory therefore extends through a sequence:

finite states;

exact orbits;

eventual equivalence;

attractor quotients;

continuous states;

approximate orbit signatures;

parameter-space partitions;

scale-dependent boundaries.

This progression connects Recursive Abiogenesis to complex dynamics and fractal formation.

The minimal finite experiment also clarifies what recursion contributes independently of geometry. Recursion produces temporal identity, attractors, basins, and eventual equivalence. Geometry adds distance, shape, continuity, and scale. Nonlinearity or branching adds the possibility of fractal boundaries and self-similar structure.

These ingredients should be studied separately before being combined. Otherwise a known fractal formula may conceal which component actually produced the observed complexity.

Recursive Abiogenesis therefore serves as the conceptual bridge between algebraic closure and dynamical emergence. It shows that repeated application alone can generate stable objects defined not by what states are now, but by what they become.

8. Complex-Plane and Fractal Abiogenesis

Complex-plane and fractal systems combine several generative ingredients that were separable in the preceding sections: algebraic operations, recursive iteration, geometric observation, scale, and boundary formation. This makes them especially important for testing whether abiogenetic discovery can move from finite exact structures into continuous or discretized dynamical spaces.

A complex number supplies both magnitude and direction. Multiplication can rotate and scale. Addition can translate. Conjugation can reflect. Nonlinear composition can fold, stretch, and repeatedly reorganize regions of the plane. Yet none of these ingredients alone produces a fractal. A single affine map usually generates only simple convergence, divergence, rotation, or periodicity. Complex recursion becomes geometrically rich only when additional structure is present, most commonly nonlinearity or branching.

The central methodological problem is therefore not simply to iterate a familiar formula. It is to identify the smallest set of primitive operations from which scale-persistent geometric objects can arise without embedding the target fractal in advance.

Complex-Plane Abiogenesis begins with anonymous transformations, exact or discretized complex states, recursive closure, and observation across multiple spatial resolutions. Fractal Abiogenesis adds the requirement that generated structure persist under refinement. Candidate systems are rejected if they collapse to a point, diverge without organization, settle into finite periodicity, fill space trivially, or produce only resolution-dependent noise. Survivors are those whose geometry remains nontrivial as depth and observational scale increase.

Two minimal routes are available. One uses branching families of affine maps. The other uses a single nonlinear recursively iterated map. Branching is the cleaner first experiment because affine generators remain simple and geometric complexity arises from recursive multiplicity rather than from a hand-selected nonlinear formula.

8.1. The minimal substrate of complex points and transformations

The basic substrate is the complex plane

$\mathbb{C} = \{x + iy : x,y \in \mathbb{R}\}$.

For exact finite computation, coefficients and seed points can be restricted initially to Gaussian integers or Gaussian rationals. A Gaussian rational has the form

a + ib,

where a and b are rational.

This restriction preserves the algebraic structure of the complex plane while permitting exact symbolic or rational arithmetic over bounded recursive depths.

The primitive objects are:

complex points;

finite sets of complex points;

complex transformations;

recursively generated orbits or point clouds.

At the beginning, no point is distinguished as geometrically meaningful. The origin may be used as a neutral seed, but it should not be treated as a special attractor unless the dynamics make it one.

The first minimal transformation family is affine:

T(z) = az + b.

The coefficient a controls rotation and scaling. The coefficient b controls translation. These interpretations need not be supplied to the discovery system. They can be inferred afterward from the action of T on selected test points.

An exact finite alphabet of coefficients might include:

0;

1 and −1;

i and −i;

1/2 and −1/2;

i/2 and −i/2;

(1+i)/2 and (1−i)/2;

small Gaussian-rational translations.

From this alphabet, anonymous maps can be enumerated and composed.

The finite coefficient alphabet is not intended as a theory of all complex geometry. It is a minimal substrate from which qualitatively different behaviors can arise:

collapse;

translation;

rotation;

reflection-like action;

contraction;

expansion;

spiral motion;

fixed points;

finite-order cycles.

The first goal is to recover these behaviors from observations rather than from map labels.

8.2. Anonymous affine maps and transformation ecology

Given a collection of affine maps

$T_j(z) = a_j z + b_j$,

the first experiment generates short compositions:

T_i;

$T_i \circ T_j$;

$T_i \circ T_j \circ T_k$;

and so forth.

Every affine composition is again affine:

$T_2(T_1(z)) = a_2 a_1 z + a_2 b_1 + b_2$.

This closure makes exact comparison possible. Two different transformation words are equivalent when they produce the same resulting coefficients.

A transformation ecology can then be built from:

composition collisions;

commuting pairs;

finite-order transformations;

common fixed points;

shared invariant sets;

contraction and expansion rates;

orbit classifications.

A map has a fixed point z^* when

$T(z^*) = z^*$.

If $a \neq 1$, the fixed point is

$z^* = b/(1-a)$.

If $a = 1$ and $b \neq 0$, no fixed point exists. If $a = 1$ and $b = 0$, every point is fixed.

These cases need not be programmed as semantic categories. The system can discover them from exact orbit observations and equation solving.

Two maps may commute:

$T_1 \circ T_2 = T_2 \circ T_1$.

Commutation defines a relation among transformations and can generate subecologies of mutually compatible actions.

Finite-order maps satisfy

$T^k = \text{identity}$

for some positive integer k. These generate periodic geometric behavior.

A transformation ecology therefore extends Algebraic Abiogenesis into geometry. The generated objects are no longer only term-function classes. They are classes of spatial actions and orbit structures.

This affine stage should precede fractal search. It establishes which geometric structures arise from composition alone and provides a baseline against which genuinely recursive scale complexity can be distinguished.

8.3. Why affine iteration alone rarely produces fractal boundaries

Consider a single affine recurrence:

$z_{n+1} = az_n + b$.

Its behavior is largely determined by $|a|$.

If $|a| < 1$, the orbit converges to a fixed point.

If $|a| > 1$, generic orbits diverge.

If $|a| = 1$, the behavior is typically rotational, translational, or periodic, depending on a and b.

These regimes can be rich enough to produce spirals, dense rotations on circles, or long periodic structures, but they do not usually create fractal boundaries from one orbit.

The reason is structural. An affine map preserves linearity. It does not fold the plane or cause different regions to stretch and recombine nonuniformly. The orbit of one point remains governed by a closed-form expression:

$z_n = a^n z_o + b(1 + a + \ldots + a^{n-1})$.

This regularity limits geometric complexity.

Therefore, a minimal fractal system requires an additional mechanism.

One possibility is nonlinearity. A nonlinear map can cause different regions of the plane to behave differently, creating interwoven basins and complicated boundaries.

The other possibility is branching. Several simple affine maps can be applied recursively, so that one state gives rise to many descendants. The complexity then arises from the combinatorial tree of compositions rather than from nonlinear distortion inside one map.

This distinction is important because it identifies two independent sources of fractal structure:

nonlinear single-orbit recursion;

branching affine recursion.

A ground-up discovery program should test them separately.

8.4. Nonlinearity as one source of recursive complexity

A nonlinear complex recurrence has the form

$z_{n+1} = F(z_n, p)$,

where p is an optional parameter.

Instead of choosing one known formula, the discovery engine can generate F from a small symbolic grammar.

A minimal grammar may contain:

the state variable z;

a parameter c;

constants such as 0, 1, −1, i, and −i;

addition;

multiplication;

optional conjugation;

optional low-complexity piecewise operations.

Candidate expressions include:

z;

c;

z + c;

zc;

z^2;

$z^2 + c$;

z(z+c);

$z^2 + z + c$.

The system should not favor one expression because it resembles a known fractal recurrence. Expressions are generated by bounded grammar complexity.

For each candidate map, the system evaluates many initial conditions or many parameter values. It records orbit signatures such as:

bounded versus escaping;

convergence;

periodicity;

transient length;

estimated Lyapunov behavior;

sensitivity to perturbation;

basin membership.

A fractal-like candidate requires a nontrivial partition of the initial-condition plane or parameter plane. If every point converges to the same attractor, there is no complex boundary. If every point escapes, the system is barren. The interesting regime contains multiple orbit classes whose boundaries remain complex under refinement.

The challenge is that the fractal object is generally not one orbit. It is a set defined by the behavior of many orbits.

For example, a parameter-space object may be defined by:

$P = \{c : \text{the orbit from } z_0 \text{ remains bounded under } F(z,c)\}$.

The observation policy creates the set. Without the boundedness distinction and parameter scan, the same recurrence would not produce a visible object.

This makes explicit that fractal objecthood depends jointly on dynamics and observation.

8.5. Branching as an alternative source of fractal structure

Branching recursion permits fractal formation even when every generator is affine.

Let

$f_1(z), \ldots, f_k(z)$

be a finite family of maps. Starting from a seed set S_0, define

$S_{n+1} = \bigcup_j f_j(S_n)$.

Each point generates k descendants at the next stage.

If the maps are contractive, the sequence of sets may approach a compact invariant set A satisfying

$A = \bigcup_j f_j(A)$.

This invariant set is the recursive object.

The branching process differs from one-map iteration. A single orbit follows one path. A branching system simultaneously realizes all words in the generator alphabet.

At depth n, the generated set contains points associated with compositions

$f_{i1} \circ f_{i2} \circ \ldots \circ f_{in}$.

The number of words grows as k^n before duplicate or overlapping outputs are removed.

This creates a direct analogy with term generation and circuit construction:

map words correspond to formal terms;

point coincidences correspond to evaluation collisions;

repeated subpatterns correspond to recursive modules;

the limit set corresponds to a stable quotient of generative histories.

Branching therefore provides the simplest bridge between Algebraic Abiogenesis and fractal geometry.

8.6. Iterated families of anonymous affine maps

The first full Fractal Abiogenesis experiment should enumerate small families of anonymous affine maps.

For two branches:

$f_o(z) = a_o z + b_o$;

$f_1(z) = a_1 z + b_1$.

The coefficients are drawn from a small exact alphabet. No prior filter should require contraction. Expanding, neutral, collapsing, and contractive systems should all be generated and then classified by behavior.

Several symmetries must be removed.

Swapping f_o and f_1 does not change the generated union.

Simultaneous translation, rotation, or scaling may produce geometrically equivalent systems under the chosen observation policy.

Complex conjugation may generate mirror-equivalent systems.

Exact duplicate maps create degenerate branching.

The recursive point sets can be generated either exhaustively or stochastically.

Exhaustive generation preserves every composition word but grows exponentially.

Random iteration samples the attractor more efficiently after contractive behavior has been identified.

For a discovery audit, shallow exhaustive depths should be combined with deeper stochastic or compressed representations.

The system should record:

map coefficients;

composition collisions;

point counts by depth;

duplicate rates;

bounding boxes;

diameter;

connected-component estimates;

occupancy grids;

symmetry signatures;

scaling estimates.

Only after these measurements are complete should the resulting attractors be compared with known iterated-function systems.

8.7. Boundedness, occupancy, and scale-dependent observation

A recursive geometric system must be converted into measurable data.

Boundedness is the first filter. If the diameter of S_n grows without control, the system may be expanding rather than attractor-forming. Some expanding systems can still produce meaningful normalized geometry, but unnormalized divergence should not be confused with stable object formation.

Collapse is the opposite failure. If all branches converge to one point or a small finite cycle, the system has recursive behavior but little geometric fertility.

Occupancy provides a spatial representation. The generated set is placed on a grid, and the number of occupied cells is counted.

At grid scale ε, let N(ε) denote the number of occupied cells. The relation between N and ε provides a measure of geometric complexity.

However, occupancy depends on:

the sampled depth;

the bounding region;

grid alignment;

point density;

numerical precision.

A credible analysis therefore uses several depths, several grid scales, and optionally several grid offsets.

The object should persist as these choices vary. A pattern visible only at one resolution is not yet a stable fractal object.

Observation must also distinguish point sets from filled regions. A dense sample may make a thin attractor appear solid. Connectivity estimates and neighborhood graphs can help prevent this misclassification.

Scale-dependent observation is therefore not merely visualization. It is the geometric analogue of enlarging a term universe or adding new countermodels.

8.8. Multi-resolution persistence and dimension estimates

Self-similar or fractal-like structure should survive refinement.

A common estimate is box-counting dimension:

$D \approx \lim \varepsilon \to 0 \, [\log N(\varepsilon)]/[\log(1/\varepsilon)]$.

In computation, the limit is replaced by a slope fitted over a finite scale range.

A candidate is more credible when:

the fitted slope is stable across recursive depths;

the slope persists across overlapping scale windows;

the estimate is robust to grid translation;

the occupied-set geometry remains nontrivial;

the result is not explained by a finite point set or a smooth curve.

Dimension alone is insufficient. Many nonfractal data sets can produce misleading slopes over narrow ranges.

Additional persistence measures may include:

component counts across scales;

hole counts;

local density distributions;

branching statistics;

symmetry recurrence;

patch similarity;

entropy of occupancy;

compression ratio of successive refinements.

A scale-stable object should not merely become more detailed. Its new detail should relate systematically to previous levels.

One possible recurrence score compares local patches at different depths after translation, rotation, and rescaling. High recurrence indicates that the same structural motif reappears at multiple scales.

This extends the concept of cross-order recurrence into geometry. Carrier order is replaced by observational scale and recursive depth.

8.9. Rejecting collapse, divergence, and geometrically barren systems

The majority of generated map families will not produce interesting fractal structure.

A rigorous pipeline should reject several barren regimes.

Exact duplicates occur when branches are identical.

Near-duplicates occur when branches generate almost the same image.

Collapse occurs when the attractor is one point or a small finite set.

Line collapse occurs when all points lie on one affine line and no further transverse structure develops.

Simple-curve behavior occurs when the set approximates a smooth arc without scale-persistent branching.

Divergence occurs when generated points escape without a stable normalized structure.

Region filling occurs when the system rapidly fills a two-dimensional area with little persistent internal organization.

Finite periodicity occurs when recursive composition produces only a bounded finite orbit.

Numerical noise occurs when apparent complexity disappears under higher precision or alternate sampling.

Transient complexity occurs when early depths appear intricate but later stages simplify.

These rejection classes should be diagnosed rather than merely discarded. Each represents a recurrent dynamical species.

Explanation subtraction can then remove known simple regimes. If a system is fully explained by common fixed-point contraction, its point cloud need not be counted as unexplained fractal fertility.

The residual ecology contains systems whose geometry cannot be reduced to these elementary explanations.

8.10. Discovering fractal attractors without selecting known formulas

A genuine Fractal Abiogenesis experiment must avoid choosing a recurrence because its output is already known.

The discovery procedure should therefore be divided into blind and interpretive phases.

During the blind phase, the system:

enumerates candidate generators;

builds recursive closures;

measures boundedness and occupancy;

tests scale persistence;

rejects barren systems;

clusters surviving geometric fingerprints;

records minimal generator descriptions.

No known fractal names or target images are used.

During the interpretive phase, surviving systems are compared with established families. Some may be rediscoveries. Others may be equivalent to known systems under affine transformations. Some may be familiar in geometry but unfamiliar in the generated parameterization. A smaller set may deserve novelty investigation.

A strong birth certificate for a recursive geometric object should contain:

the exact generators;

the recursive rule;

the observation policy;

point sets or occupancy grids by depth;

scale-persistence statistics;

dimension estimates with uncertainty;

symmetry reductions;

perturbation tests;

comparison against simpler explanations;

evidence of recurrence across nearby systems or alternate descriptions.

The methodological achievement would not be that a computer drew an intricate image. It would be that a minimal generator ecology produced a stable geometric object whose objecthood was certified across depth, scale, and perturbation without the target being prescribed.

The larger progression is:

complex points
→ anonymous transformations
→ composition and recursion
→ orbit or set generation
→ spatial observation
→ scale-persistent equivalence
→ attractor formation
→ canonical geometric object.

This progression suggests a minimal fractal toolbox:

a geometric substrate;

a small map alphabet;

nonlinearity or branching;

recursive closure;

an observational partition;

multi-resolution persistence;

degeneracy rejection;

canonical generator recovery.

Fractal Abiogenesis thus occupies an important intermediate position. It is more geometric and less exact than finite algebra, yet more controlled and auditable than unrestricted visual search. It tests whether the general theory of object formation can survive the passage from finite equality to continuous-scale structure.

9. Cellular Automaton Abiogenesis

Cellular automata provide one of the clearest environments in which recursive object formation can be studied from the ground up. They combine a finite local state alphabet, a spatial lattice, an anonymous local rule, and repeated synchronous updating. From these minimal ingredients, they can generate fixed configurations, oscillations, traveling patterns, expanding fronts, chaotic regions, conserved quantities, and interacting computational structures.

The principal attraction of cellular automata is that the local rule can be completely specified and exhaustively enumerated at small scales. A binary one-dimensional automaton with a radius-one neighborhood has only 256 possible local update rules. Every rule can therefore be tested without selecting a preferred example. This creates a natural analogue of the exhaustive finite operation spaces used in Algebraic and Logical Abiogenesis.

The central methodological question is not which known automata produce interesting pictures. It is whether persistent objects, interaction laws, and computational modules can emerge from anonymous local rules before familiar classifications are applied. A moving pattern becomes an object when successive configurations differ only by translation. An oscillator becomes an object when it returns after a fixed number of steps. A collision becomes a rule when the same incoming structures repeatedly produce the same outgoing structures. A conservation law emerges when a measurable quantity remains invariant under the local update across many configurations.

Cellular Automaton Abiogenesis therefore studies the progression:

anonymous local rule
→ recursive spatial evolution
→ recurrent patterns
→ equivalence under translation or period
→ stable localized objects
→ collision laws
→ computational modules.

9.1. Cellular automata as spatially distributed recursion

A cellular automaton consists of a lattice of cells, a finite state alphabet, a neighborhood definition, and a local update rule.

Let the lattice be indexed by integers. Each cell at position i and time t has state

$c_i(t) \in S$,

where S is a finite alphabet.

For a binary automaton,

$S = \{0,1\}$.

A radius-one local rule uses the states of a cell and its immediate neighbors:

$c_i(t+1) = R(c_{i-1}(t), c_i(t), c_{i+1}(t))$.

The same rule is applied simultaneously at every lattice position.

This is recursion distributed over space. The global configuration C(t) is updated by a transformation

$C(t+1) = F_r(C(t))$,

where F_r is induced by the local rule R.

The rule is local, but its consequences are global. Information can propagate across the lattice because each update depends on neighboring states, and those neighbors depended on their own neighbors at the previous step.

Unlike a unary map on a finite carrier, the state space of a cellular automaton grows exponentially with lattice size and becomes infinite on an unbounded lattice. Yet the local rule space remains finite and often small enough for exhaustive study.

This separation between a tiny rule and an enormous configuration space makes cellular automata ideal for studying emergence. Complex behavior cannot be attributed to a large explicit program. It must arise through recursive interaction among repeated local updates.

9.2. Anonymous local update rules

The first experiment should treat every local rule anonymously.

For a binary radius-one automaton, each neighborhood contains three bits. There are eight possible neighborhood configurations:

000, 001, 010, 011, 100, 101, 110, 111.

A rule assigns one output bit to each neighborhood. Since each of the eight outputs may be zero or one, there are

$2^8 = 256$

possible rules.

The usual decimal numbering of these rules should be retained only as an indexing convenience, not as a semantic label. The discovery process should not privilege historically famous rules or known universality results.

Each rule is evaluated across a standardized ecology of initial conditions, including:

single active cells;

short finite patterns;

random configurations at several densities;

periodic backgrounds;

localized perturbations of periodic backgrounds;

pairs of separated candidate structures;

finite rings of different sizes.

The output should be a behavioral fingerprint rather than a name.

This fingerprint may record:

time to extinction;

time to temporal periodicity;

spatial period;

growth rate of active regions;

entropy over time;

compressibility;

sensitivity to one-cell perturbations;

presence of localized recurrent structures;

propagation velocities;

collision outcomes;

predecessor counts;

candidate conserved quantities.

Rules related by simple transformations should be identified. Reflection of the neighborhood, interchange of zero and one, and combinations of these symmetries can produce equivalent global behavior under relabeling. Failure to remove such duplicates would inflate recurrence.

The anonymous rule ecology should therefore be organized by behavioral equivalence before traditional categories are considered.

9.3. Exhaustive binary radius-one rule ecologies

The complete set of 256 elementary binary rules is small enough to support exhaustive first-stage analysis.

For each rule, the system can run a fixed battery of initial conditions and produce a structured record of its evolution. The result is not one visual space-time diagram but a multidimensional behavioral profile.

The ecology may be represented as a graph. Each node corresponds to a rule. Edge weights reflect similarity between behavioral fingerprints.

Several kinds of recurrence may then appear.

Some rules rapidly eliminate all finite disturbances.

Some freeze configurations into fixed spatial patterns.

Some generate short temporal cycles.

Some produce expanding periodic structures.

Some amplify small perturbations into irregular regions.

Some support localized patterns moving through a background.

Some conserve quantities such as parity or weighted state count.

The purpose of the ecology is not to reproduce one preexisting classification. It is to discover clusters from measured behavior and then compare those clusters with known taxonomies afterward.

As in Algebraic Abiogenesis, raw richness must be interpreted carefully. A rule producing visually dense output may have low structural fertility if the output is essentially incompressible noise. A visually simple rule may support exact conservation laws, long transients, or reusable moving structures.

The relevant score should therefore combine complexity with persistence, compressibility, and interaction.

9.4. Fixed, periodic, chaotic, and propagating regimes

Several broad dynamical regimes can emerge from the anonymous rule ecology.

A fixed regime reaches a configuration C satisfying

$F_r(C) = C$.

A periodic regime reaches a cycle:

$F_r^p(C) = C$

for some positive period p.

A propagating regime contains a pattern P that reappears after p time steps shifted by d cells:

$F_r^p(P) = T^d(P)$,

where T^d denotes spatial translation.

A chaotic or irregular regime exhibits sustained sensitivity, high pattern diversity, and weak short-description compression.

These categories should not be assigned solely by visual inspection. Each should have measurable criteria.

Fixed behavior can be detected exactly on finite lattices.

Periodicity can be certified by state recurrence.

Propagation can be detected by matching future configurations under translation.

Irregularity can be estimated through growth of distinct local blocks, perturbation spread, entropy, and compression ratio.

Many rules exhibit mixtures. A localized propagating structure may move through a periodic background while emitting irregular debris. A rule may be ordered for sparse initial conditions and chaotic for dense ones.

Therefore, the proper unit of classification may not be the rule alone but the pair:

rule plus initial-condition family.

This mirrors the role of designated sets in logic. The same local operation can support different global objects under different observational and environmental policies.

9.5. Translation-equivalence and moving-pattern objecthood

Pointwise equality is too strict to recognize a traveling object.

Suppose a localized pattern shifts two cells to the right after four time steps. Its exact coordinates change, but its internal form and dynamical role persist.

Define an equivalence relation:

$P \sim Q$

when Q is a spatial translation of P.

For temporally evolving patterns, define:

$P \sim_r Q$

when there exist integers $p > 0$ and d such that

$F_r^p(P) = T^d(Q)$.

A pattern satisfying

$F_r^p(P) = T^d(P)$

is a recurring propagating object.

Its velocity is

$v = d/p$.

The object is not any one configuration at one coordinate. It is the equivalence class of translated temporal phases.

This is a clear example of object formation through quotienting. Many distinct lattice configurations become one moving object because position and phase are treated as presentation details.

Additional quotient policies are possible.

Reflection-equivalence identifies left-moving and right-moving mirror variants.

Phase-equivalence identifies different stages of one oscillator.

Background-equivalence identifies the same localized disturbance embedded at different positions in a periodic environment.

These policies must be stated explicitly. If translation is quotiented away, velocity remains as a dynamical property while absolute position disappears.

Moving-pattern objecthood therefore arises when recursive evolution preserves form modulo a symmetry.

9.6. Localized structures as emergent computational modules

A localized structure is a finite or spatially bounded deviation from a background that persists over time.

Persistence may take several forms:

stationary stability;

temporal oscillation;

translation;

periodic deformation;

growth with stable boundary structure.

A candidate module should satisfy more than longevity. It should also possess reproducible interactions or reusable behavior.

The discovery process can begin by scanning space-time diagrams for recurrent finite patches. Candidate patches are extracted and reinserted into standardized backgrounds. Their subsequent evolution is tested across positions, phases, and small perturbations.

A stable localized structure should reproduce its characteristic behavior independently of where it is placed, provided the background is compatible.

Once identified, the structure can be assigned a canonical description containing:

support width;

temporal period;

translation velocity;

phase states;

background requirements;

predecessor patterns;

interaction ports;

robustness to perturbation.

A localized object becomes a computational module when its presence, absence, phase, position, or collision can encode and transform information.

This is directly analogous to NAND-circuit motif formation. Individual cell updates correspond to primitive gates. A persistent space-time pattern corresponds to a reusable subcircuit. A collision network corresponds to a larger circuit assembled dynamically rather than statically.

9.7. Pattern collisions and interaction laws

Two localized structures can be placed on trajectories that cause them to meet. Their collision may produce:

annihilation;

reflection;

transmission;

fusion;

fission;

phase change;

new moving structures;

stationary residue;

irregular debris.

A collision law is established when the same incoming object classes, phases, velocities, and separations reliably produce the same outgoing object classes.

The proper description is not one space-time picture but a transition rule:

$A + B \rightarrow C + D$.

Here A, B, C, and D denote equivalence classes of localized patterns rather than exact coordinate configurations.

Collision analysis requires systematic variation of:

relative phase;

initial separation;

background phase;

orientation;

velocity;

small perturbations.

A rule that appears deterministic at one phase may have several outcomes at others.

The collision ecology can be represented as a directed hypergraph. Incoming object sets point to outgoing object sets.

This hypergraph may reveal:

closed families of interacting particles;

catalytic structures that survive collisions;

signals that route or transform other signals;

collision-based logic;

memory elements;

reversible interactions;

particle production and decay.

The emergence of stable interaction laws is a higher level of object formation. Individual moving patterns become entities, and repeated collisions become an effective physics governing those entities.

9.8. Conservation laws and reversibility

A conservation law is a quantity Q such that

$Q(C(t+1)) = Q(C(t))$

for every configuration in a specified domain.

Candidate quantities may include:

number of active cells;

parity of active cells;

weighted sums over neighborhoods;

moment-like spatial quantities;

block counts;

additive local densities;

topological or symbolic invariants.

The search for conservation laws should not assume which quantity matters. A library of low-complexity local observables can be generated and tested across all configurations of bounded rings.

For a ring of size n, exhaustive configuration testing is possible for modest n. If a candidate law survives increasing ring sizes, it gains credibility.

Conservation may also be approximate or conditional. A quantity may be preserved only on a recurrent subspace or during collisions among particular localized objects.

Reversibility is related but distinct. A global automaton is reversible when every configuration has exactly one predecessor.

On finite rings, reversibility can be tested by checking whether the global update map is bijective.

Nonreversible rules may still contain reversible subdynamics. A family of localized objects and their collisions may evolve bijectively even though the surrounding rule loses information elsewhere.

This distinction is important for emergent computation. A globally dissipative substrate may support locally reversible modules.

Conservation and reversibility provide strong filters for mathematical fertility because they support predictive laws extending beyond visual pattern recognition.

9.9. Compression, predictability, and computational fertility

Cellular automaton output can be complex in several different senses.

A configuration may have high entropy but low structural organization.

Another may be highly periodic and therefore easily compressed.

A third may combine regular backgrounds with sparse moving defects.

The most fertile regime may lie between trivial order and unstructured noise.

Compression can be measured through:

run-length encoding;

dictionary compression;

block entropy;

grammar size;

predictive model complexity;

description length of recurrent objects and collisions.

A rule is computationally fertile when a compact set of discovered modules and interaction laws explains a large fraction of its evolution.

For example, a long space-time diagram may be compressed into:

a periodic background;

three propagating object classes;

two oscillator classes;

a table of collision outcomes.

This description is much more informative than the raw cell array.

Explanation subtraction can then be applied. Once periodic background behavior is accounted for, residual complexity is attributed to localized structures. Once known objects are removed, unexplained debris can be examined for additional modules.

A rule producing visually intricate but incompressible output may have low consequence reserve. It generates complexity without reusable structure.

A fertile rule supports:

stable objects;

recurrent interactions;

hierarchical modules;

predictive compression;

counterfactual manipulation;

composition into larger computations.

These criteria parallel the standards used for algebraic theories, logical calculi, and circuits.

9.10. From anonymous rules to emergent machines

The long-term objective is to discover machines rather than merely patterns.

A machine is a stable arrangement of components that transforms inputs into outputs through reproducible internal dynamics.

In a cellular automaton, candidate machines may include:

signal generators;

signal absorbers;

reflectors;

routers;

memory cells;

counters;

oscillators;

logic gates;

arithmetic modules;

self-repairing structures;

constructors.

The discovery process should not begin by requesting these objects. It should first identify persistent patterns, interaction laws, and reusable assemblies.

Higher-order search can then operate over discovered components rather than individual cells.

The developmental hierarchy becomes:

cell state;

local transition;

recurrent patch;

localized object;

collision law;

component assembly;

machine;

ecology of machines.

This hierarchy resembles the progression from NAND gates to motifs, adders, and larger circuits. The difference is that cellular components exist in time and space. Wiring is replaced by trajectories and collisions.

A machine can be certified by demonstrating:

a finite component description;

stable operation across multiple placements;

repeatable input-output behavior;

robustness to phase variation;

modularity;

reuse in several larger systems;

compression advantage over raw cell-level description.

An especially strong result would be the autonomous discovery of a cellular structure whose function was not specified in advance but whose repeated behavior supports a compact post hoc interpretation.

The broader significance lies in the relationship between local law and emergent ontology. The microscopic rule contains no explicit particles, signals, gates, or machines. These appear only after recursive evolution produces patterns stable under translation, temporal recurrence, and interaction.

Cellular Automaton Abiogenesis therefore provides a model of how a world of objects can arise from uniform local updating.

Its minimal toolbox consists of:

a finite state alphabet;

a lattice;

a neighborhood;

an anonymous local rule;

recursive synchronous evolution;

symmetry-aware observation;

pattern equivalence;

collision analysis;

compression and stability filters.

The field also offers a bridge toward neural systems. A neural network is not usually a homogeneous lattice, but both systems transform distributed states through repeated local or layered interactions. Persistent activation motifs, routing patterns, and causal circuits may play roles analogous to cellular objects.

The central lesson is that recursive local rules can generate new levels of description. At the microscopic level there are only cell states and updates. At the emergent level there can be moving objects, laws of collision, information channels, and machines.

The transition between these levels is not metaphorical. It can be formalized through observational equivalence, quotienting, recurrence, and compression. Cellular automata therefore supply one of the most concrete laboratories for studying the birth of effective objects from a minimal recursive substrate.

10. Neural Abiogenesis: Training as Ground-Up Object Formation

Neural networks are normally described as systems that learn functions by adjusting parameters to reduce error. This description is correct but incomplete. It explains optimization without explaining how internal objects arise. Features, concepts, roles, circuits, and algorithms are often identified after training, yet the developmental process by which they become stable remains poorly understood.

Neural Abiogenesis treats training as a generative process of object formation. The model begins with distributed numerical states rather than mature concepts. Repeated exposure to data, recursive parameter updates, and pressure to perform under finite capacity progressively reorganize those states. Distinctions that matter for prediction are retained. Distinctions that do not matter are merged, ignored, or made interchangeable. Stable internal objects emerge when many different inputs, activation patterns, or computational paths become equivalent with respect to their downstream effect.

This interpretation extends Algebraic Abiogenesis. In the algebraic setting, formal terms are identified when they induce the same function. In the neural setting, internal states may be identified when substituting one for another preserves behavior across a specified context family. The resulting equivalence classes need not correspond to individual neurons or directions. They may be distributed across layers, attention heads, activation subspaces, or recurrent computational trajectories.

Training is therefore not only parameter fitting. It may be the progressive construction of functional quotients. Compression, pruning, distillation, and bottlenecks can then reveal or strengthen these quotients by removing distinctions the model no longer needs.

10.1. Neural networks as recursively composed operational systems

A neural network is a composition of transformations. For a feedforward model,

$h_0 = x$,

$h_{k+1} = T_k(h_k)$,

and

$y = T_{l-1} \circ \dots \circ T_1 \circ T_0(x)$.

Each layer transforms one representation into another. The model as a whole is therefore a recursively composed operational system.

The primitive operations may include:

linear transformations;

nonlinear activation functions;

normalization;

attention;

convolution;

routing;

residual addition;

memory access;

recurrent state updating.

The architecture specifies how these operations may be composed, but it does not specify the semantic objects that will inhabit the intermediate states. Those objects must emerge through training.

At initialization, internal representations are usually weakly structured relative to the task. Activation directions, attention patterns, and hidden states may vary greatly without supporting stable interpretation.

As training proceeds, certain transformations become recurrently useful. A layer may begin to separate inputs according to a task-relevant distinction. An attention head may route information between related positions. A hidden-state direction may encode a variable needed by several later computations. A circuit may become responsible for a reusable operation.

These structures arise from composition under selection. Their emergence parallels the formation of algebraic terms and circuit modules. Repeated transformations generate many candidate internal states, but only some acquire stable downstream roles.

The model should therefore be analyzed not as a static function alone but as an ecology of composed internal operations.

10.2. Optimization as developmental recursion

Training adds a second form of recursion.

Let θ_t denote the parameters at training step t. An update rule U produces:

$\theta_{t+1} = U(\theta_t, B_t, L_t)$,

where B_t is a data batch and L_t represents loss or gradient information.

The model interpreting the next batch is modified by the current batch. Training is therefore a recursive developmental process.

This recursion differs from ordinary forward inference. Forward recursion transforms an input through a fixed parameter state. Developmental recursion transforms the parameter state itself.

The two processes interact:

the current parameters determine which representations are produced;

those representations determine the loss;

the loss determines the parameter update;

the updated parameters change the next representations.

This feedback can stabilize internal structures. A representation that improves many predictions may receive repeated reinforcement. A redundant distinction may receive no selective support and gradually disappear.

The developmental trajectory may contain identifiable transitions. A feature may first appear weakly, then sharpen, then become causally necessary, and finally survive compression. A circuit may form from several initially independent components that become coordinated over training.

Neural Abiogenesis therefore asks not only what structures exist in the final model but when they were born.

A developmental birth point may be defined as the earliest checkpoint at which a candidate object satisfies specified stability criteria:

reliable behavioral role;

recurrence across examples;

causal importance;

persistence under continued training;

resistance to compression;

reappearance across random seeds.

This converts training history into an object-formation record.

10.3. Data distinctions and task-relevant equivalence

Training data contains far more variation than any model can preserve exactly. Inputs differ in wording, order, style, noise, context, and incidental detail. Yet many of these differences are irrelevant to the task.

A predictive model succeeds by learning which distinctions matter.

Suppose two inputs x and x′ produce the same task-relevant outcome. The model may benefit from representing them similarly, but geometric similarity is not the fundamental requirement. The important condition is behavioral interchangeability.

Define a task-relative relation:

$x \sim_t x'$

when replacing x by x′ does not alter the relevant prediction or action under the task policy.

This relation may be coarse or fine.

For image classification, many pixel-level differences may be ignored.

For translation, word order and grammatical structure may remain essential.

For theorem proving, two syntactically different expressions may be interchangeable if they have the same mathematical role.

For language modeling, the same token may require different internal treatment depending on context.

Task-relative equivalence is therefore contextual. Two states may be equivalent for one downstream objective and distinct for another.

Training can be viewed as learning a hierarchy of such relations. Early layers may preserve local surface distinctions. Later layers may merge inputs according to broader semantic or functional roles.

A concept emerges when many different observations are mapped into a stable class that supports the same downstream transformations.

10.4. Functional substitutability of internal states

Geometric similarity is often used to compare neural representations. Activations may be clustered by Euclidean distance, cosine similarity, or linear projection. These measures are useful but do not directly establish identity of function.

Two activation vectors may be close yet have different downstream consequences. Conversely, two distant vectors may be functionally interchangeable because later layers map them to the same behavior.

A stronger criterion is substitution.

Let h and h′ be internal states at the same intervention site. Define

$h \sim h'$

when replacing h with h′ across a family of downstream contexts produces equivalent outputs.

The context family may include:

different continuations;

different prompts;

different surrounding tokens;

different tasks;

different perturbations;

different later-layer states.

Substitutability should be tested causally. The activation is replaced, patched, ablated, or projected, and the downstream effect is measured.

Exact equality is unrealistic in large neural systems. A tolerance or policy is required. Two states may be equivalent when their output distributions differ by less than a threshold, when their ranking of actions is unchanged, or when all task-relevant decisions remain the same.

This produces a policy-indexed equivalence:

$h \sim_p h'$,

where P specifies the observational contexts and tolerated behavioral differences.

Stable neural objects should survive reasonable changes in P. If an apparent feature disappears as soon as the context family broadens, it may be an artifact of narrow observation.

Functional substitutability is the neural counterpart of term-function equality and truth-vector equivalence.

10.5. Activations as representatives of behavioral equivalence classes

An activation vector should not automatically be identified with a concept. It is one presentation of a possible functional class.

The same internal role may be realized by many numerical states. Context, layer normalization, residual streams, and distributed coding can produce different activation patterns that later computation treats similarly.

A neural object is therefore better represented as an equivalence class:

$[h] = \{h' : h' \sim_p h\}$.

The class may contain states generated by:

different surface forms;

different languages;

different tasks;

different positions;

different layers;

different model widths;

different training seeds.

The stronger the recurrence across these conditions, the more credible the object.

This view also explains polysemanticity. A single neuron may participate in several equivalence classes depending on context. The neuron is not itself the object. It is one coordinate used by several distributed objects.

Similarly, one concept may not be localized to a single neuron. It may be realized by a subspace, a sparse combination of features, or a circuit involving multiple layers.

Quotienting by behavioral equivalence shifts the unit of analysis from hardware coordinates to functional role.

This can produce more stable interpretability. Individual parameters may change under fine-tuning or reparameterization while the functional class remains.

10.6. The birth of internal objects during training

To study neural object formation, checkpoints must be treated as developmental stages.

At each checkpoint, the system records:

activation responses;

downstream substitution effects;

feature selectivity;

causal intervention results;

effective rank;

circuit connectivity;

compression sensitivity;

cross-context recurrence.

A candidate object may begin as a diffuse tendency. Several activation patterns partially predict the same downstream behavior but are not yet interchangeable.

As training continues, the class may sharpen. Within-class substitution causes smaller behavioral change, while substitution across classes causes larger change.

This suggests a measurable objecthood score combining:

within-class substitutability;

between-class separability;

cross-context stability;

causal importance;

compression survival;

recurrence across seeds.

The birth point is not necessarily one exact optimization step. It may be a transition interval during which the equivalence relation becomes stable.

Several developmental patterns are possible.

Gradual birth occurs when a representation steadily sharpens.

Sudden birth occurs when a circuit or feature appears after a phase transition.

Competitive birth occurs when several candidate representations emerge but one becomes dominant.

Layer migration occurs when a function shifts from one region of the network to another.

Fusion occurs when previously separate features collapse into one reusable abstraction.

Fission occurs when one coarse feature divides into specialized subfeatures.

These are neural analogues of theory splitting, cluster merging, and recursive peeling.

10.7. Features, concepts, and algorithms as quotient structures

Different kinds of internal neural objects may correspond to different forms of quotienting.

A feature identifies inputs sharing one property relevant to prediction.

A concept identifies a broader class supporting multiple related transformations.

A role identifies states that occupy the same computational position even if their semantic content differs.

An algorithm identifies a recurrent sequence of transformations that maps varying inputs through the same procedural structure.

For example, a network may represent:

syntactic subjecthood;

negation;

numerical magnitude;

spatial containment;

entity continuity;

quotation context;

causal direction;

intermediate arithmetic state.

These objects need not be explicitly labeled during training. They become useful because they compress recurring predictive structure.

An algorithmic object is especially important. It is not merely a static activation class but a trajectory class.

Two internal computational traces may be equivalent when they implement the same input-output transformation under different surface conditions.

This suggests extending neural equivalence from states to paths:

$\pi \sim \pi'$

when two internal trajectories have the same downstream functional role.

A learned algorithm may therefore be a quotient of many computational traces.

10.8. Reusable neural circuits and modular computation

A neural circuit is a set of components whose coordinated causal activity implements a function.

The components may include:

attention heads;

neurons;

features;

residual-stream directions;

routing mechanisms;

memory positions;

recurrent states.

A circuit becomes module-like when it is reused across inputs or tasks.

Reuse is central because it distinguishes a stable computational object from a one-off activation coincidence.

The NAND experiments provide an exact analogy. Repeated gate patterns became XOR-like motifs. Those motifs were reused inside larger arithmetic circuits. Shared reuse reduced total circuit cost.

In neural models, a reusable circuit may lower representational and computational burden by applying one learned transformation across many contexts.

Candidate modules should be tested through interventions:

disable the circuit;

replace it with an equivalent implementation;

activate it in a new context;

reroute its output;

compress its components;

measure downstream loss.

A genuine module should possess a degree of functional autonomy. Its input-output role should persist even when incidental implementation details change.

This does not imply that neural computation is perfectly modular. Modules may overlap, share components, or interact dynamically. Objecthood can be partial and policy-relative.

The research goal is to detect when distributed activity supports a sufficiently stable interface to count as a reusable computational object.

10.9. Recurrence across seeds, widths, and architectures

A feature discovered in one model may be an accident of initialization or architecture. Recurrence provides stronger evidence.

Train multiple models with:

different random seeds;

different widths;

different depths;

different optimization schedules;

different data orders;

different architectures;

different compression regimes.

Then search for functionally corresponding internal structures.

Direct parameter comparison is usually impossible because neural representations can be permuted, rotated, or distributed differently. Alignment should therefore be based on behavior.

Two internal structures recur when they support the same intervention-defined function across models.

Cross-model recurrence can be measured by:

shared response profiles;

equivalent substitution effects;

matching causal dependencies;

common task transfer;

similar compression survival;

interchangeability after learned alignment.

A neural object that recurs across architectures is especially significant. It may reflect a stable requirement of the task rather than a peculiarity of one model design.

This is analogous to cross-order recurrence in Algebraic and Logical Abiogenesis. Carrier size becomes model width or architecture. Stable law becomes stable internal computation.

10.10. Neural objecthood as causal rather than geometric identity

The deepest implication is that neural objecthood should be defined causally.

A geometric cluster may look coherent but have no effect on model behavior. A direction may correlate with a concept without being used by the model. An attention pattern may appear interpretable while being causally replaceable.

Causal identity asks what the state does.

A candidate neural object should satisfy several conditions.

It should be interventionally coherent: replacing representatives within the class should preserve downstream behavior.

It should be discriminative: replacing a representative with one from a different class should alter behavior in predicted ways.

It should be reusable: the same class should participate in several contexts.

It should be stable: its role should persist across training checkpoints or compression.

It should be compressible: many presentations should admit a shorter shared description.

It should support lawful composition: combining neural objects should produce recurrent transformations or outputs.

Under this view, a concept is not a label attached to an activation. It is a stable causal quotient of internal computation.

The full developmental sequence becomes:

unstructured parameters
→ recurrent activation patterns
→ functional substitutability
→ equivalence classes
→ reusable circuits
→ compressed internal objects
→ lawful composition among objects.

Neural Abiogenesis therefore reframes representation learning. The model does not merely store external objects. It constructs internal object systems by repeatedly collapsing distinctions that have no predictive consequence.

This interpretation also changes the role of compression. Compression is not only an engineering procedure applied after intelligence has formed. It may be one of the mechanisms by which internal objecthood becomes possible.

A finite-capacity model must compress its experience. To compress successfully, it must discover which differences can be forgotten without loss. Stable objects are the classes that remain after those differences have been erased.

The resulting hypothesis is:

intelligence develops by constructing functional quotients under recursive optimization.

This hypothesis is experimentally testable. It predicts that internal object classes should become more stable during training, recur across independently trained models, survive compression, support causal substitution, and organize into reusable computational circuits.

If these predictions hold, Neural Abiogenesis would connect mathematical object formation with representation learning at a mechanistic level. Algebraic terms, logical formulas, circuit graphs, cellular patterns, and neural states would all become instances of one general process:

many generated presentations become one object when they remain interchangeable under the transformations and observations that matter.

11. Model Compression as Quotient Formation and Theory Peeling

Model compression is usually treated as an engineering problem. A trained network is considered too large, too slow, too expensive, or too difficult to deploy, and compression methods are applied to reduce memory, computation, energy use, or latency. Parameters are pruned, numerical precision is lowered, weight matrices are factorized, activations are bottlenecked, and smaller models are trained to imitate larger ones. Success is measured by how much size or computation can be removed without unacceptable loss of performance.

Neural Abiogenesis gives compression a deeper interpretation. Compression is the controlled removal of distinctions. Two parameters, activations, features, circuits, or models become equivalent when replacing one with the other leaves the selected behavior unchanged. Every successful compression therefore provides evidence that the original representation contained distinctions that were not essential under the observation policy.

This is structurally similar to quotient formation. A large representational space is partitioned into classes whose members are functionally interchangeable. The compressed model retains one effective representative of each relevant class and discards redundant presentations. Compression can also operate recursively. After one layer of redundancy has been removed, new equivalences may become visible. This resembles theory peeling in Algebraic Abiogenesis, where the consequences of a discovered law are subtracted before the residual structure is analyzed.

The central question is therefore not only how much a model can be compressed. It is what equivalence relation each compression method reveals, which internal objects survive, and whether the surviving structure becomes more canonical, modular, and causally intelligible.

11.1. Compression as removal of distinctions without behavioral loss

Let M be a trained model and C(M) a compressed version. The ordinary objective is to make C(M) smaller while preserving an acceptable approximation to the behavior of M.

This can be written schematically as:

minimize Complexity(C(M))

subject to

Distance(Behavior(M), Behavior(C(M))) ≤ ε.

The behavioral distance depends on the observation policy. It may measure:

prediction accuracy;

difference between output distributions;

ranking consistency;

calibration;

task reward;

robustness;

internal activation similarity;

performance across a family of interventions.

A compression is therefore always relative to what is being preserved. Two models may be equivalent for classification but not for explanation, uncertainty estimation, adversarial robustness, or transfer learning.

This policy dependence mirrors object formation in the earlier fields. In logic, designated values determine which semantic distinctions matter. In circuits, truth-vector equality determines whether two netlists compute the same function. In neural compression, the evaluation policy determines which differences may be erased.

Suppose two parameter settings θ and θ' produce indistinguishable behavior over a task family T. Define:

$\theta \sim_t \theta'$

when

$M\theta(x) \approx M\theta'(x)$

for all relevant x in T.

The quotient of parameter space by this relation identifies many numerical implementations with one functional model.

The same idea can be applied locally. Two neurons may be equivalent if exchanging them, together with the necessary compensating transformation, leaves the model unchanged. Two activation directions may be equivalent if downstream computation uses only their shared projection. Two subcircuits may be equivalent if they implement the same context-dependent transformation.

Compression discovers or imposes these equivalences by reducing representational degrees of freedom. The important distinction is between numerical removal and functional identification. Deleting a parameter is not inherently meaningful. It becomes meaningful when the deletion demonstrates that a family of parameter variations belongs to one behavioral class.

Compression can therefore be interpreted as an empirical quotient operation:

many internal presentations
→ one preserved behavioral role.

11.2. Pruning and parameter equivalence

Pruning removes weights, neurons, attention heads, channels, or other components judged unnecessary.

The simplest method removes individual weights with small magnitude. More structured methods remove entire rows, columns, channels, heads, blocks, or computational paths. Iterative pruning alternates between removal and retraining, permitting the remaining model to absorb lost function.

From the quotient perspective, a pruned parameter is one whose variation within a certain range does not create a behaviorally relevant distinction. If changing a weight to zero leaves the model's outputs unchanged under the observation policy, the original nonzero value and zero belong to the same approximate functional class.

This does not mean that all small weights are unimportant or that all large weights are essential. Parameters interact. A weight may be individually dispensable but collectively necessary with others. Two redundant pathways may each appear removable until both are deleted.

The relevant equivalence therefore often applies to parameter sets rather than isolated scalars.

Let S be a set of components and M\S the model with those components removed. The set is behaviorally null under policy P when:

$\text{Behavior}_P(M) \approx \text{Behavior}_P(M \setminus S)$.

A null component set lies inside the kernel of the model's task-relative causal map.

Pruning can reveal several forms of redundancy.

Duplicate redundancy occurs when several components perform nearly the same function.

Backup redundancy occurs when one component compensates for the removal of another.

Distributed redundancy occurs when a function is spread across many individually weak components.

Conditional redundancy occurs when a component matters only in rare contexts.

Developmental redundancy occurs when a structure was useful during training but is no longer needed after another circuit has formed.

These distinctions matter for Neural Abiogenesis. A component that can be removed from the final model may still have played a formative role. Compression performed only after training could miss developmental objects that scaffolded later structures.

Checkpoint-specific pruning can therefore test when a component becomes necessary, redundant, or replaceable. The temporal pattern may reveal theory-like peeling within training. Early representations explain part of the behavior, are later absorbed into more efficient structures, and then disappear.

Structured pruning is particularly useful for detecting modules. If an entire head, channel, or block can be removed without loss, it may represent redundant implementation. If a compact collection must remain jointly intact, it may form a causally coherent circuit.

Pruning thus converts necessity into an experimental property. What survives repeated removal is a candidate component of the model's minimal functional core.

11.3. Quantization and numerical quotienting

Quantization reduces the number of distinct numerical values used to represent weights or activations.

A high-precision parameter space contains enormous numerical variation. Yet the model may not use most of those distinctions. Mapping many nearby values to one shared level tests whether they are behaviorally interchangeable.

A quantizer Q partitions a numerical domain into bins. Values w and w′ satisfy

$w \sim_Q w'$

when

$Q(w) = Q(w')$.

This is an explicit quotient of the numerical representation.

The important question is whether this numerical quotient respects the model's functional structure. If performance remains stable, the distinctions removed by Q were largely unnecessary under the task policy.

Quantization can reveal scale-dependent identity. Two values may be distinct at 32-bit precision but equivalent at 8-bit precision. Further compression may show that 4-bit or even binary distinctions are sufficient in some regions of the model.

Different layers may tolerate different quotient coarseness. A uniform quantization policy may destroy sensitive computations while preserving redundant ones. Adaptive quantization can therefore measure where numerical precision carries genuine functional information.

Activation quantization is especially informative. It reduces the number of internal states available during inference. If many continuous activation patterns collapse into a small set of effective codes without loss, the model may already be operating through discrete or nearly discrete internal objects.

This suggests a direct experiment. Record activation states before and after quantization. Then test whether quantized states correspond to stable functional classes under downstream substitution.

If they do, quantization has not merely approximated the computation numerically. It has exposed a latent symbolic or finite-state structure.

Quantization can also produce artificial equivalences that were not present in the original model. Excessive coarsening may merge states with different downstream roles. The result is then not a faithful quotient but a destructive collapse.

A good quantization certificate should therefore include:

the numerical partition;

the behavioral policy;

the loss introduced by each merge;

the contexts in which the merge fails;

the stability of the resulting classes across checkpoints and seeds.

Quantization becomes scientifically interesting when the compressed levels align with causal computational roles.

11.4. Low-rank factorization and redundant directions

Large neural layers often use matrices whose nominal dimensionality exceeds their effective functional rank.

Suppose a weight matrix W is approximated by

$W \approx UV$,

where U and V have a smaller intermediate dimension r.

The factorization replaces a high-dimensional transformation with a passage through an r-dimensional bottleneck.

This can be interpreted as quotienting the input space by directions that W does not distinguish strongly. If two input vectors differ only along a direction in or near the null space of W, they produce the same or nearly the same output.

Formally, for an exact linear map W, define:

$x \sim_W y$

when

$W(x) = W(y)$.

Then x and y are equivalent exactly when their difference lies in the kernel of W.

The effective object is not the original input vector but its coset modulo the kernel.

Low-rank compression extends this idea approximately. Small singular directions contribute little to the output under the measured data distribution. Removing them identifies states that differ primarily in those directions.

This connects neural compression directly to quotient geometry.

The singular spectrum can also reveal developmental change. During training, a matrix may begin with many comparable directions, then concentrate function into a smaller subspace. Effective rank reduction may mark the birth of a compressed internal representation.

However, low rank alone does not establish meaningful object formation. A matrix can be low rank for trivial or accidental reasons. The retained directions must be tested for:

causal relevance;

cross-context stability;

recurrence across models;

interpretability of downstream roles;

survival under further compression.

Low-rank factorization may also expose compositional stages. A layer may first project inputs into a small quotient space and then transform the resulting classes. The intermediate coordinates could correspond to latent objects or computational variables.

The factorization should therefore be examined as more than a matrix approximation. It may be a structural decomposition:

fine-grained representation
→ quotient coordinates
→ task-specific transformation.

11.5. Distillation and behavioral equivalence between models

Knowledge distillation trains a smaller student model to reproduce the behavior of a larger teacher.

The student may imitate:

hard labels;

teacher probability distributions;

intermediate representations;

attention patterns;

reasoning traces;

selected internal circuits.

Distillation is a direct experiment in model-level equivalence. It asks how much of the teacher's implementation can be discarded while preserving a chosen behavioral surface.

If a small student reproduces the teacher over a broad context family, many differences between their architectures and parameters become irrelevant under that policy. The teacher and student are distinct presentations of one approximate functional object.

Define:

$M_1 \sim_P M_2$

when both models are behaviorally interchangeable under policy P.

Distillation searches for a simpler representative of the equivalence class [M].

This is analogous to selecting a minimal NAND netlist for a truth function. The function is the equivalence class; the circuit is one implementation. A distilled model may serve as a canonical or lower-complexity representative of a larger model's behavior.

The analogy is not exact because neural behavior is evaluated over enormous or open-ended domains. Distillation may preserve common cases while failing on rare contexts. The equivalence certificate must therefore specify the evaluation distribution and uncertainty.

Distillation can reveal which teacher structures are essential.

If the student reproduces outputs but not internal representations, the teacher's internal geometry may be implementation-specific.

If independently distilled students converge on similar internal circuits, those circuits may reflect stable task structure.

If a behavior cannot be distilled below a certain capacity, the required internal object system may have a measurable minimum complexity.

Multi-stage distillation resembles recursive theory peeling. A large model is compressed into a smaller one, then compressed again. Structures that survive every stage are candidates for the irreducible behavioral theory of the task.

Distillation can also separate creation from revelation. A student may invent a new internal representation that is more compact than the teacher's. In that case, compression does not merely reveal the teacher's objects. It constructs an alternative quotient system with the same external behavior.

11.6. Bottlenecks and forced abstraction

A bottleneck restricts the amount of information that can pass between stages of a model.

The bottleneck may limit:

dimensionality;

number of tokens;

number of active experts;

memory capacity;

communication bandwidth;

precision;

sparsity;

latent code length.

Because the model cannot preserve every distinction, it must decide which information is relevant for future prediction.

This makes bottlenecks natural laboratories for object formation.

Suppose an encoder maps inputs x to latent codes z:

$z = E(x)$,

and a decoder or downstream system uses z to perform a task.

If two inputs receive the same code,

$E(x) = E(x')$,

then the encoder has identified them. The latent code represents an equivalence class of inputs under the learned policy.

A useful bottleneck must preserve enough information for the task while removing irrelevant variation. It therefore approximates a minimal sufficient quotient.

The severity of the bottleneck can be varied. With high capacity, the model may memorize fine distinctions. As capacity decreases, classes merge. Some merges preserve behavior; others cause failure.

This produces a refinement sequence of learned ontologies.

At one capacity, every example may remain distinct.

At a lower capacity, surface variants merge.

At a still lower capacity, related concepts merge.

Below a critical threshold, task-relevant distinctions collapse.

The sequence can be analyzed as a lattice of representational resolutions.

A stable object is one that persists across a range of bottleneck strengths. A fragile object appears only at one narrow capacity. A fundamental task distinction survives until near the performance-collapse threshold.

Bottleneck experiments can therefore identify the minimal distinctions required by a task.

They may also produce unexpected abstractions. The model may discover a compressed variable not supplied in the labels because that variable efficiently organizes many predictions.

Such a latent object is especially important for Neural Abiogenesis. It is created by compression pressure rather than directly named by supervision.

11.7. Compression as creation versus revelation of structure

Two interpretations of compression must be distinguished.

Under the revelation view, training has already formed stable internal objects. Compression removes redundant parameters and exposes their minimal implementation.

Under the creation view, compression pressure actively causes the model to form abstractions that would not otherwise exist.

Both processes are plausible.

Post-training pruning often behaves like revelation. The model's behavior is already established, and unnecessary implementation detail is removed.

Training with a severe bottleneck behaves more like creation. The model must organize information into reusable classes because it lacks capacity for a less structured solution.

Quantization-aware training may occupy an intermediate position. Numerical constraints shape the representations during learning, causing the model to adapt to the quotient imposed by the hardware.

Distillation can also create structure. A student may solve the imitation problem using a representation unlike the teacher's but more compressed and modular.

The distinction can be tested experimentally by comparing several developmental paths:

train a large model and compress it afterward;

train a small model directly;

train a large model under continuous sparsity pressure;

train through a bottleneck from the beginning;

alternate expansion and compression;

distill repeatedly through decreasing capacities.

If the same functional objects emerge across these paths, they are likely stable attractors of the task.

If objects appear only under compression pressure, compression is a creative force.

If an object appears in the large model and survives every reduction, compression is revealing a preexisting core.

If different compressed models implement the same behavior through different objects, functional objecthood may be more stable than representational objecthood.

These experiments distinguish task-imposed structure from architecture-imposed structure and compression-induced structure.

11.8. Measuring which objects survive compression

A compression study should track internal objects, not only final accuracy.

Suppose a candidate object O has been identified in the uncompressed model through functional substitution, causal intervention, or recurrent activation structure.

After compression, search for a corresponding object O'.

Correspondence should be defined behaviorally rather than by parameter location. The compressed model may move or redistribute the computation.

An object-survival score may combine:

preservation of input selectivity;

preservation of downstream causal effect;

substitutability across models;

preservation of composition with other objects;

recurrence across contexts;

similar contribution to task performance;

persistence of associated inference or circuit behavior.

Several outcomes are possible.

Exact survival occurs when the same functional object remains with a smaller implementation.

Distributed survival occurs when the role persists but is spread across different components.

Fusion occurs when several objects merge into one compressed representation.

Fission occurs when one broad object is replaced by several specialized components.

Migration occurs when the object moves to a different layer or pathway.

Extinction occurs when the object disappears and the behavior is lost or implemented by a fundamentally different mechanism.

These outcomes provide a developmental taxonomy of neural compression.

The strongest candidate objects are those that survive multiple compression methods. A structure preserved under pruning, quantization, low-rank factorization, and distillation is less likely to be an artifact of one representation.

Survival should also be tested across task distributions. An object may appear dispensable on the original benchmark but become essential under distribution shift or compositional generalization.

A complete compression certificate should therefore specify not only what was removed but which functional equivalence classes remained.

11.9. Detecting the developmental birth point of an internal quotient

If training constructs functional quotients, their formation should be observable over time.

Let $\sim_t$ denote a functional equivalence relation measured at training checkpoint t.

Early in training, the relation may be unstable. States grouped together at one checkpoint may separate later. Alternatively, examples initially represented distinctly may gradually become interchangeable.

The birth of a quotient occurs when a partition of internal states becomes sufficiently stable and causally meaningful.

A practical procedure is:

record checkpoints frequently;

collect internal states for a fixed context family;

measure downstream substitution effects;

construct equivalence classes at each checkpoint;

align classes across checkpoints;

track class persistence, merging, and splitting;

test causal and compression survival.

A quotient birth score might increase when:

within-class behavioral variation decreases;

between-class behavioral variation increases;

class assignments stabilize over time;

the class becomes reused across tasks;

the corresponding circuit becomes causally necessary;

the class survives later compression.

This makes object birth a measurable transition rather than a retrospective interpretation.

The developmental history may reveal that internal objects do not emerge in isolation. One quotient may enable another. A model may first form token-level classes, then syntactic roles, then semantic relations, then reusable reasoning procedures.

This produces a dependency graph of object formation.

Theory peeling can be applied temporally. Once an early object explains part of the model's behavior, subtract its contribution and analyze the residual. New structures may become visible only after dominant features are accounted for.

The model's development can then be described as a sequence:

initial undifferentiated representations;

first stable equivalence classes;

composition of classes into circuits;

compression of circuits into modules;

higher-order quotient formation.

This is the neural analogue of theories emerging layer by layer from finite operation tables.

11.10. Compression as a microscope for mechanistic interpretability

Mechanistic interpretability seeks to explain how a model computes. Large models make this difficult because function is distributed across vast numbers of parameters and activations.

Compression can act as a microscope by removing redundant degrees of freedom.

A smaller equivalent model has fewer candidate mechanisms. Sparse structure reveals causal pathways. Low-rank structure reveals active subspaces. Quantized activations reveal discrete internal states. Distillation can produce a cleaner implementation of the same behavior.

However, compression is not automatically explanatory. A compressed model may remain opaque. It may also replace an interpretable distributed mechanism with a smaller but less transparent one.

The interpretive value of compression depends on whether the compressed units align with stable functional objects.

A promising workflow is:

identify behaviorally equivalent states;

compress within each equivalence class;

extract minimal causal circuits;

test replacement and reuse;

express the surviving mechanism in symbolic or modular form;

verify the explanation through intervention.

This workflow parallels the NAND-circuit calibration. Truth-function equivalence supports exact circuit minimization. Repeated motifs become modules. Larger systems are then described in terms of those modules.

For neural systems, exact truth vectors are unavailable, but intervention-defined behavioral signatures can serve a similar role.

The long-term goal is a quotient-aware interpretability system that reports:

which distinctions the model preserves;

which distinctions it ignores;

which internal states are interchangeable;

which circuits implement each equivalence class;

which objects survive compression;

which transformations govern composition among objects.

Such a system would not merely label neurons. It would reconstruct the model's internal ontology.

Compression then becomes more than deployment technology. It becomes an experimental procedure for discovering what the model considers the same.

The broader hypothesis can be stated precisely:

a neural object is a stable functional equivalence class whose representatives can be compressed into a reusable causal mechanism without loss of task-relevant behavior.

This definition unifies representation, objecthood, and compression.

Algebraic Abiogenesis identifies expressions under equal term functions. Logical Abiogenesis identifies formulas under mutual consequence. Circuit Abiogenesis identifies graphs under equal truth functions. Neural Abiogenesis identifies internal states and mechanisms under downstream behavioral substitutability.

Model compression operationalizes the identification. It tests which distinctions can be removed and which must remain.

The resulting developmental sequence is:

training generates representational diversity;

functional equivalence identifies redundant distinctions;

compression collapses equivalent presentations;

theory peeling exposes residual structure;

recurrent modules become canonical internal objects;

higher-order objects form through composition and reuse.

If this sequence can be measured across checkpoints and reproduced across models, compression will become evidence for a ground-up process of object construction inside artificial intelligence.

The model's reduced size would then be only the visible engineering outcome. The deeper event would be ontological: a large space of possible internal states has been reorganized into a smaller system of stable functional objects.

12. A Unified Research Program for Minimal Toolboxes

The preceding sections describe a family of experiments that appear different on the surface but share a common generative architecture. Algebra begins with finite carriers and anonymous operations. Logic adds designated values and consequence. Circuits add graph reuse and multi-output coordination. Recursive systems add temporal iteration. Complex dynamics add

geometry and scale. Cellular automata add distributed local updating. Neural systems add learning, functional substitution, and compression.

These domains can therefore be organized as a unified research program centered on minimal toolboxes for object formation.

A minimal toolbox is not simply a software library or list of algorithms. It is the smallest collection of primitives required to let a field generate its own candidate objects from below. It includes a substrate, anonymous generators, closure rules, observation policies, equivalence tests, stability filters, and certification procedures. A successful toolbox does not merely reproduce familiar examples. It supports blind generation, retrospective interpretation, adversarial rejection, and eventual discovery of structures not supplied as targets.

The long-term objective is to build a family of interoperable discovery systems capable of producing theories, proof calculi, circuits, attractors, geometric objects, cellular machines, and neural concepts through one common logic of recursive generation and quotient formation.

The unifying sequence is:

primitive distinctions
→ anonymous transformations
→ recursive closure
→ observational equivalence
→ compression
→ stable object classes
→ laws of composition
→ higher-order modules.

The central scientific question is whether this sequence is merely an analogy across fields or a genuinely general mechanism of mathematical and computational object formation.

12.1. Minimal field-specific substrates and generators

Every field requires a substrate and a generator alphabet.

The substrate determines what can be distinguished initially. The generators determine how new candidate structures may be formed. Minimality is important because an overly rich starting language can hide the source of the discovered structure.

In algebra, the minimal substrate is a finite carrier and the minimal generator is one anonymous operation of fixed arity. Formal terms provide closure.

In logic, the substrate consists of finite values and designated subsets. Anonymous operations generate formulas. Preservation of designation generates consequence.

In circuits, the substrate consists of input truth functions. NAND or another functionally complete primitive generates new signals through graph composition.

In recursion, the substrate is a state space and the generator is a unary update map.

In complex dynamics, the substrate is the complex plane and the generator may be an affine or nonlinear transformation.

In fractal systems, the substrate may be points or compact sets, while the generators are branching transformations or nonlinear recurrences.

In cellular automata, the substrate is a lattice configuration and the generator is a local update table.

In neural learning, the substrate consists of data, parameters, activations, and computational contexts. The generators are layers, update rules, and intervention-defined transformations.

A field-specific toolbox should begin with the smallest generator family capable of nontrivial closure.

For algebra, one binary operation may be sufficient.

For logic, one binary operation plus designatedness may be sufficient for initial consequence formation.

For circuits, one NAND gate is sufficient for all finite Boolean functions.

For recursion, one unary map is sufficient for cycles and basins.

For fractals, one nonlinear map or two affine branches may be sufficient.

For cellular automata, one local binary rule is sufficient.

For neural systems, the minimal generator remains less clear. It may be one trainable linear transformation plus nonlinearity, one recurrent cell, one attention block, or a small compositional network.

The correct minimal toolbox is therefore itself a discovery problem. A generator set is minimal when removing any primitive destroys a class of stable structures that cannot be recovered through composition of the remainder.

12.2. Algebra, logic, circuits, recursion, geometry, and automata

The six principal computational domains form a natural developmental sequence.

Algebra studies equality under composition.

Logic studies preservation under designation.

Circuits study implementation under behavioral equivalence.

Recursion studies identity through future evolution.

Geometry studies identity under spatial transformation and scale.

Cellular automata study objecthood under distributed recursive dynamics.

Each domain adds one new ingredient to the previous architecture.

Algebra supplies formal composition and congruence.

Logic adds asymmetry between premises and conclusions.

Circuits add reuse through directed acyclic graphs.

Recursion adds temporal dependence.

Geometry adds metric, topology, and resolution.

Cellular automata add spatial distribution and interaction among emergent objects.

Neural systems combine nearly all of these features. They use compositional transformations, directional inference, distributed circuits, developmental recursion, high-dimensional geometry, and compression.

This sequence suggests that the toolboxes should be implemented modularly. A general term engine can support algebra and logic. A semantic quotient engine can support circuits and neural states. A recursive orbit engine can support finite dynamics, complex maps, and cellular automata. A symmetry and canonicalization engine can operate across all fields.

The larger platform would not treat each field as a separate application. It would reuse common mechanisms while allowing domain-specific observation policies.

12.3. Shared criteria for stable object formation

A candidate structure should not become an object merely because it appears once.

Stable object formation requires multiple independent criteria.

The first is recurrence. The structure should appear across models, seeds, scales, carrier orders, or environments.

The second is behavioral substitutability. Different presentations should be interchangeable under the selected policy.

The third is closure. The equivalence class should remain meaningful under the generators of the field.

The fourth is compression. Replacing many presentations with one class or module should reduce description or implementation cost.

The fifth is causal coherence. Intervention on the object should produce predictable changes.

The sixth is robustness. Small perturbations should not destroy the object unless fragility is intrinsic to its behavior.

The seventh is consequence reserve. The object should support further predictions, laws, compositions, or constructions.

The eighth is cross-context persistence. A structure that exists only under one narrow test may be an artifact of the observation policy.

The ninth is canonicalizability. The class should admit a stable representative, normal form, minimal circuit, attractor description, or compressed module.

The tenth is adversarial survival. Explicitly constructed counterexamples should fail to dissolve the claimed structure.

These criteria can be combined into an objecthood score, but a single scalar should not replace the full certificate. Different objects may be stable in different ways. A theorem, a moving cellular pattern, and a neural feature require different forms of evidence.

The important point is that objecthood becomes testable rather than intuitive.

12.4. Audit certificates and reproducible discovery pipelines

Every discovered structure should be accompanied by a machine-readable birth certificate.

For an algebraic theory, the certificate should include:

the operation tables;

the term universe;

the collision set;

the inferred basis;

the explanation-subtraction results;

the countermodels;

the quotient structure;

the normal-form procedure.

For a logic, it should include:

the matrices;

the designated subsets;

the formula universe;

the consequence fingerprint;

the reducedness tests;

the inferred proof rules;

the soundness and completeness audit.

For a circuit, it should include:

the primitive gate;

the input functions;

the search strategy;

the truth vectors;

the minimum netlists;

the lower-bound certificate;

the discovered motifs;

the multi-output relations.

For recursive systems, it should include:

the maps;

the seeds;

the orbit decomposition;

the equivalence policy;

the attractor and basin structure.

For fractal candidates, it should include:

the generators;

the recursive depth;

the occupancy data;

the scaling analysis;

the perturbation tests;

the rejection of simpler explanations.

For cellular automata, it should include:

the local rule;

the initial-condition ecology;

the detected patterns;

the translation and period classes;

the collision table;

the conservation tests.

For neural systems, it should include:

the checkpoints;

the context family;

the interventions;

the functional equivalence classes;

the compression path;

the circuit correspondence;

the recurrence across seeds and architectures.

A birth certificate should record not only successful evidence but failed hypotheses. Rejected rules, failed cluster thresholds, nonrobust structures, and counterexamples are part of the scientific result.

This is especially important for open-ended discovery. Without a complete audit trail, a system may produce attractive structures that cannot be distinguished from selection bias or retrospective storytelling.

12.5. Distinguishing calibration, rediscovery, and genuine novelty

The research program must maintain a strict distinction among three outcomes.

A calibration reproduces a known structure to validate the methodology. The ternary algebraic law, implication-like logical cluster, NAND XOR motif, half-adder, full-adder, and multiplexer are examples of calibration when their identities and implementations are already established.

A rediscovery occurs when the system finds a known result without using it as a target. Rediscovery is stronger than calibration because the structure was not selected in advance, but it is not mathematical novelty.

A novel discovery is a structure, law, circuit, logic, or object not already present in the relevant literature and supported by independent verification.

Methodological novelty is separate from object novelty. A pipeline can be new even when its first outputs are known. Conversely, a new object can be found by a conventional method.

A credible novelty claim requires:

blind generation;

full documentation of the search space;

evidence that the result was not encoded in the objective;

comparison with existing literature;

independent reproduction;

adversarial validation;

clear separation between discovery and interpretation.

The program should therefore use a maturity ladder.

Level 1: illustrative example.

Level 2: exact reproduction.

Level 3: cross-order or cross-scale recurrence.

Level 4: adversarial certification.

Level 5: autonomous basis or module extraction.

Level 6: prediction beyond the discovery sample.

Level 7: independent replication.

Level 8: literature-supported novelty.

This ladder prevents calibration results from being overstated while preserving their methodological value.

12.6. Cross-field transfer of discovered modules

A central advantage of the unified architecture is that discoveries in one field may become tools in another.

An algebraic normal-form engine can be used to simplify logical formulas.

A logical consequence engine can evaluate whether circuit modules support implication-like behavior.

Circuit minimization can provide models for neural module compression.

Recursive orbit analysis can detect state machines inside cellular automata or neural systems.

Fractal scaling tools can analyze activation geometry or parameter-space boundaries.

Cellular automaton pattern detection can be adapted to recurring activation motifs in spatiotemporal neural networks.

Neural compression methods can be applied to large algebraic or circuit search spaces to identify low-dimensional structure.

This transfer should not be purely metaphorical. The software representations should be interoperable.

A generated object can be represented by:

a carrier or state space;

a generator set;

an observation policy;

an equivalence relation;

a canonical representative;

a composition interface.

This common representation would allow a circuit module, proof rule, attractor, or neural object to be stored in the same high-level format.

The system could then ask whether one object can implement or simulate another.

A circuit may realize a logical connective.

A cellular collision may realize a circuit gate.

A neural circuit may approximate an algebraic operation.

A recursive system may generate a symbolic logic.

Cross-field equivalence could reveal that different substrates support the same abstract object.

12.7. Semantic and syntactic routes to object birth

The research program should pursue two complementary directions.

The semantic route begins with models or behavior:

finite operation tables;

logical matrices;

truth vectors;

orbit observations;

activation interventions.

Equivalence is discovered from shared behavior, and compact rules are inferred afterward.

The syntactic route begins with generative transformations:

rewrite rules;

proof steps;

program composition;

circuit construction;

recursive grammars.

Objects are formed by identifying derivations or expressions that normalize to the same result.

The semantic route is:

models
→ equivalence
→ rules.

The syntactic route is:

rules
→ derivations
→ equivalence classes.

A strong theory should connect both routes.

In logic, semantic consequence should correspond to derivability.

In algebra, term-function equality should correspond to rewriting or proof from a basis.

In circuits, truth-function equality should correspond to transformation among netlists.

In neural systems, behavioral equivalence should correspond to interchangeable internal computation.

Convergence between independently generated semantic and syntactic structures would be powerful evidence of object stability.

One system could infer rules from behavior, while another generates behavior from rules. Agreement would reduce the possibility that the result is an artifact of one direction of construction.

12.8. Autonomous theorem, rule, circuit, and concept discovery

Once stable objects and operations exist, the system can begin a second stage of discovery.

In algebra, it can search for theorems that compress the consequence structure.

In logic, it can search for lemmas, admissible rules, and normal proof forms.

In circuits, it can search for reusable multi-output modules and new minimum implementations.

In recursion, it can search for invariants and classification theorems.

In complex dynamics, it can search for parameter relations predicting orbit regimes.

In cellular automata, it can search for collision laws and higher-order machines.

In neural systems, it can search for causal concepts, reusable algorithms, and modular internal interfaces.

Theorem or module value should not be measured only by novelty of syntax. Candidate discoveries can be scored by:

number of consequences explained;

reduction in description length;

frequency of reuse;

independence from existing results;

predictive reach;

robustness across models;

ability to generate new objects;

importance in proofs or circuits.

This creates a discovery process that does not merely enumerate consequences. It identifies structural bottlenecks whose recognition reorganizes the field.

A theorem becomes significant when it compresses many derivations.

A circuit becomes significant when it reduces many implementations.

A concept becomes significant when it organizes many neural states.

A module becomes significant when it recurs across systems.

This unifies mathematical importance with compression and reuse.

12.9. Experimental tests of Neural Abiogenesis

Neural Abiogenesis requires direct experimental validation.

The first experiment should use a small but nontrivial model trained on a task containing compositional structure. The model should be large enough to form internal circuits but small enough for frequent checkpointing and intervention.

The experimental pipeline should:

train multiple seeds;

record dense checkpoint sequences;

collect internal activations;

define a family of downstream intervention contexts;

measure functional substitutability;

construct equivalence classes at each checkpoint;

track class birth, fusion, fission, and extinction;

apply pruning, quantization, low-rank compression, and distillation;

test which classes survive;

search for recurrent causal circuits;

compare structures across widths and architectures.

The task should not explicitly label the internal objects being sought.

Suitable initial domains include:

finite symbolic transformations;

small algorithmic tasks;

compositional language fragments;

graph operations;

arithmetic with intermediate state;

cellular automaton prediction;

circuit simulation.

The strongest evidence would be the spontaneous formation of a reusable internal object that:

was not named in the loss;

recurs across seeds;

appears at a measurable training stage;

supports causal substitution;

survives compression;

participates in several tasks;

admits a compact circuit description.

A second experiment should compare compression paths.

One model is trained large and compressed afterward.

One is trained small from the start.

One is trained under continuous sparsity pressure.

One is trained through a bottleneck.

One is repeatedly expanded and recompressed.

If the same objects recur, they are likely imposed by the task. If different objects produce the same behavior, the task may support multiple internal ontologies.

12.10. Mathematical objects and intelligent representations as stable functional quotients

The unified hypothesis is that both mathematical objects and intelligent representations are stable functional quotients.

A mathematical expression becomes an object when many presentations are identified under lawful equivalence.

A proposition becomes an object when many formulas occupy the same inferential role.

A circuit function becomes an object when many netlists compute the same behavior.

An attractor becomes an object when many trajectories share the same asymptotic fate.

A cellular particle becomes an object when many translated space-time configurations represent one persistent structure.

A neural concept becomes an object when many internal states are interchangeable under downstream computation.

In every case, the object is not identical to one physical or syntactic presentation.

The object is the class that remains invariant under the transformations that matter.

This viewpoint connects abstraction with controlled forgetting. An object forms when irrelevant differences are removed while relevant behavior is preserved.

The process is therefore both constructive and compressive.

Generation creates many candidate presentations.

Observation determines which differences matter.

Equivalence identifies substitutable presentations.

Compression collapses them.

Closure turns the resulting classes into a new operational world.

This last step is essential. A quotient becomes mathematically fertile when its classes can themselves be combined, transformed, and used to generate further structure.

The birth of objects is therefore recursive. Objects formed at one level become primitives at the next.

NAND gates form XOR motifs.

XOR motifs form adders.

Adders form arithmetic systems.

Cell patterns form moving particles.

Particles form collision machines.

Neural features form circuits.

Circuits form algorithms.

The hierarchy of abstraction is built by repeated quotient formation.

12.11. Limitations and unresolved questions

The program faces substantial conceptual and computational limitations.

The first is combinatorial explosion. Term spaces, circuit graphs, operation tables, formula universes, and neural intervention spaces grow rapidly. Exact enumeration is possible only at small scales.

The second is observation-policy dependence. Different policies produce different objects. A general theory must explain how policies are selected, compared, and refined.

The third is approximate equivalence. Finite algebra and circuits permit exact equality. Geometry and neural systems require tolerance. Approximate equivalence may fail transitivity and produce unstable quotient structures.

The fourth is novelty verification. Open-ended systems may rediscover known results under unfamiliar notation. Literature comparison remains difficult and partly human.

The fifth is semantic interpretation. A stable quotient may be mathematically real without having an obvious human meaning.

The sixth is scale transfer. Structures found in small finite systems may not persist in larger systems.

The seventh is computational bias. Search order, grammar design, sampling, and scoring can privilege certain structures.

The eighth is proof. Computational recurrence is not always equivalent to a general theorem.

The ninth is causal certification. Correlation and compression do not automatically establish functional objecthood.

The tenth is open-ended growth. A system that continually promotes discovered modules may eventually become difficult to audit.

These limitations do not invalidate the framework. They define the standards required for its development.

A mature platform will need:

exact methods where possible;

formal uncertainty where exactness is impossible;

multiple observation policies;

independent search routes;

countermodel generation;

symmetry reduction;

reproducible logs;

human and machine novelty audits;

theorem-proving integration;

causal intervention.

12.12. Conclusion: Compression, equivalence, and the construction of worlds

The research program developed here begins from one reversal.

Mathematics need not always begin after its objects have been selected.

A system can begin with primitive distinctions and anonymous transformations. Recursive composition generates an ecology of possibilities. Observation identifies which differences affect behavior. Stable equivalence classes form quotients. Compression selects canonical representatives. Reuse and closure transform those classes into higher-order objects.

Algebraic Abiogenesis demonstrates the process through finite operations and term identities.

Logical Abiogenesis extends it to consequence and proof rules.

Circuit Abiogenesis extends it to exact function classes, shared subgraphs, and arithmetic modules.

Recursive Abiogenesis extends it to orbits, cycles, and attractors.

Fractal Abiogenesis extends it to scale-persistent geometry.

Cellular Automaton Abiogenesis extends it to moving objects, collisions, and emergent machines.

Neural Abiogenesis proposes that learning and compression perform the same process internally.

The common mechanism is not raw complexity. It is selective collapse.

Many generated states are treated as one because their differences no longer matter under a policy.

That collapse is not destruction when it preserves function. It is abstraction.

Every stable object is therefore both a loss and a gain.

Fine distinctions are lost.

A reusable world of classes, laws, and transformations is gained.

This suggests a general definition:

an object is a stable equivalence class of generated presentations whose members are substitutable under a specified family of transformations and observations.

The definition applies to expressions, propositions, functions, trajectories, geometric sets, cellular patterns, and neural states.

The larger implication is that intelligence and mathematics may not merely describe worlds composed of objects. They may construct object worlds through the same recursive operation of compression under equivalence.

A trained model may appear smaller after pruning, quantization, or distillation. Yet the deeper event may not be reduction in file size. It may be the consolidation of an internal ontology.

A finite operation table may appear to contain only entries. Yet term composition and stable collision can reveal a theory.

A NAND gate may appear to contain only one Boolean operation. Yet recursive reuse can produce arithmetic.

A local cellular rule may appear to contain no particles or machines. Yet repeated updating can create both.

The minimal toolboxes proposed here are therefore not merely computational conveniences. They are instruments for studying how new levels of reality become mathematically available.

The final ambition is a general science of object birth:

a science that identifies the minimum substrate required for a field;

generates candidate structure without naming the target;

discovers stable equivalence;

compresses presentations into objects;

certifies their recurrence and fertility;

and promotes those objects into the primitives of a new level.

Such a science would connect theorem discovery, logic formation, circuit invention, dynamical emergence, geometric construction, cellular computation, and neural representation learning.

It would also provide a concrete form of Mathematical It from Bit.

The bit is not sufficient by itself.

The essential process is:

information
→ operation
→ recursion
→ equivalence
→ compression
→ object.

When the object becomes a new generator, the process begins again.

13. References

1. Youvan, Douglas C. “Mathematical It from Bit: Algebraic Abiogenesis, Anonymous Finite Operations, and the Automated Birth of Abstract Algebra.” July 2026. DOI: 10.13140/RG.2.2.25987.26403.
2. Youvan, Douglas C. “Algebraic Abiogenesis.” Source code, computational audits, and methodological documentation. GitHub repository, 2026. https://github.com/DougYouvan/algebraic-abiogenesis
3. Shannon, Claude E. “A Mathematical Theory of Communication.” *Bell System Technical Journal* 27 (1948): 379–423, 623–656.
4. Wheeler, John Archibald. “Information, Physics, Quantum: The Search for Links.” In *Complexity, Entropy, and the Physics of Information*, edited by Wojciech H. Zurek, 3–28. Redwood City, California: Addison-Wesley, 1990.
5. Landauer, Rolf. “Irreversibility and Heat Generation in the Computing Process.” *IBM Journal of Research and Development* 5, no. 3 (1961): 183–191. DOI: 10.1147/rd.53.0183.
6. Kolmogorov, Andrey N. “Three Approaches to the Quantitative Definition of Information.” *Problems of Information Transmission* 1, no. 1 (1965): 1–7.
7. Chaitin, Gregory J. “On the Length of Programs for Computing Finite Binary Sequences.” *Journal of the ACM* 13, no. 4 (1966): 547–569. DOI: 10.1145/321356.321363.
8. Rissanen, Jorma. “Modeling by Shortest Data Description.” *Automatica* 14, no. 5 (1978): 465–471. DOI: 10.1016/0005-1098(78)90005-5.
9. Birkhoff, Garrett. “On the Structure of Abstract Algebras.” *Proceedings of the Cambridge Philosophical Society* 31, no. 4 (1935): 433–454. DOI: 10.1017/S0305004100013463.
10. Burris, Stanley, and H. P. Sankappanavar. *A Course in Universal Algebra*. Graduate Texts in Mathematics 78. New York: Springer-Verlag, 1981.
11. Grätzer, George. *Universal Algebra*. 2nd ed. New York: Springer, 2008. DOI: 10.1007/978-0-387-77487-9.
12. McKenzie, Ralph, George McNulty, and Walter Taylor. *Algebras, Lattices, Varieties*. Vol. 1. Monterey, California: Wadsworth & Brooks/Cole, 1987.
13. Ebbinghaus, Heinz-Dieter, and Jörg Flum. *Finite Model Theory*. 2nd ed. Berlin: Springer, 2005.

14. Baader, Franz, and Tobias Nipkow. *Term Rewriting and All That*. Cambridge: Cambridge University Press, 1998. DOI: 10.1017/CBO9781139172752.

15. Knuth, Donald E., and Peter B. Bendix. “Simple Word Problems in Universal Algebras.” In *Computational Problems in Abstract Algebra*, edited by John Leech, 263–297. Oxford: Pergamon Press, 1970.

16. Bolan, Matthew, et al. “The Equational Theories Project: Advancing Collaborative Mathematical Research at Scale.” arXiv:2512.07087, 2025.

17. Berlioz, Clément, and Paul-André Melliès. “The Latent Space of Equational Theories.” arXiv:2601.20759, 2026.

18. Blok, Willem J., and Don Pigozzi. *Algebraizable Logics*. Memoirs of the American Mathematical Society 77, no. 396. Providence, Rhode Island: American Mathematical Society, 1989. DOI: 10.1090/memo/0396.

19. Czelakowski, Janusz. *Protoalgebraic Logics*. Dordrecht: Kluwer Academic Publishers, 2001. DOI: 10.1007/978-94-017-2807-2.

20. Font, Josep Maria, and Ramon Jansana. *A General Algebraic Semantics for Sentential Logics*. Lecture Notes in Logic 7. Berlin: Springer-Verlag, 1996.

21. Tarski, Alfred. *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*. 2nd ed. Edited by John Corcoran. Indianapolis: Hackett Publishing, 1983.

22. Troelstra, Anne S., and Helmut Schwichtenberg. *Basic Proof Theory*. 2nd ed. Cambridge: Cambridge University Press, 2000. DOI: 10.1017/CBO9781139168717.

23. Negri, Sara, and Jan von Plato. *Structural Proof Theory*. Cambridge: Cambridge University Press, 2001. DOI: 10.1017/CBO9780511527340.

24. Avigad, Jeremy. “Proof Theory.” arXiv:1711.01994, 2017.

25. Shannon, Claude E. “The Synthesis of Two-Terminal Switching Circuits.” *Bell System Technical Journal* 28, no. 1 (1949): 59–98.

26. Knuth, Donald E. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. Upper Saddle River, New Jersey: Addison-Wesley, 2011.

27. Haaswijk, Winston, Mathias Soeken, Alan Mishchenko, and Giovanni De Micheli. “SAT Based Exact Synthesis Using DAG Topology Families.” In *Proceedings of the 55th Annual Design Automation Conference*, Article 53, 1–6. New York: Association for Computing Machinery, 2018. DOI: 10.1145/3195970.3196111.

28. Haaswijk, Winston, Mathias Soeken, Alan Mishchenko, and Giovanni De Micheli. “SAT-Based Exact Synthesis: Encodings, Topology Families, and Parallelism.” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, no. 4 (2020): 871–884. DOI: 10.1109/TCAD.2019.2897703.

29. Reichl, Matthias, Friedrich Slivovsky, and Stefan Szeider. “Circuit Minimization with QBF-Based Exact Synthesis.” *Proceedings of the AAAI Conference on Artificial Intelligence* 37, no. 4 (2023): 4087–4094. DOI: 10.1609/aaai.v37i4.25524.

30. Devaney, Robert L. *An Introduction to Chaotic Dynamical Systems*. 3rd ed. Boca Raton, Florida: CRC Press, 2022. DOI: 10.1201/9780429280801.

31. Milnor, John. *Dynamics in One Complex Variable*. 3rd ed. Annals of Mathematics Studies 160. Princeton, New Jersey: Princeton University Press, 2006.

32. Hutchinson, John E. “Fractals and Self-Similarity.” *Indiana University Mathematics Journal* 30, no. 5 (1981): 713–747. DOI: 10.1512/iumj.1981.30.30055.

33. Mandelbrot, Benoit B. *The Fractal Geometry of Nature*. San Francisco: W. H. Freeman, 1982.

34. Falconer, Kenneth. *Fractal Geometry: Mathematical Foundations and Applications*. 3rd ed. Chichester: Wiley, 2014.

35. Barnsley, Michael F. *Fractals Everywhere*. 2nd ed. Boston: Academic Press, 1993.

36. Wolfram, Stephen. “Statistical Mechanics of Cellular Automata.” *Reviews of Modern Physics* 55, no. 3 (1983): 601–644. DOI: 10.1103/RevModPhys.55.601.

37. Toffoli, Tommaso, and Norman Margolus. *Cellular Automata Machines: A New Environment for Modeling*. Cambridge, Massachusetts: MIT Press, 1987.

38. Ilachinski, Andrew. *Cellular Automata: A Discrete Universe*. Singapore: World Scientific, 2001. DOI: 10.1142/4702.

39. Cook, Matthew. “Universality in Elementary Cellular Automata.” *Complex Systems* 15, no. 1 (2004): 1–40. DOI: 10.25088/ComplexSystems.15.1.1.

40. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, Massachusetts: MIT Press, 2016.

41. Tishby, Naftali, Fernando C. Pereira, and William Bialek. “The Information Bottleneck Method.” In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, 368–377. 1999. arXiv/0004057.

42. Shwartz-Ziv, Ravid, and Naftali Tishby. “Opening the Black Box of Deep Neural Networks via Information.” arXiv:1703.00810, 2017.

43. Saxe, Andrew M., Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D. Tracey, and David D. Cox. “On the Information Bottleneck Theory of Deep Learning.” In *International Conference on Learning Representations*. 2018. arXiv:1802.08734.

44. Achille, Alessandro, and Stefano Soatto. “Emergence of Invariance and Disentanglement in Deep Representations.” *Journal of Machine Learning Research* 19, no. 50 (2018): 1–34.

45. LeCun, Yann, John S. Denker, and Sara A. Solla. “Optimal Brain Damage.” In *Advances in Neural Information Processing Systems 2*, 598–605. San Mateo, California: Morgan Kaufmann, 1990.

46. Denton, Emily L., Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. “Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation.” In *Advances in Neural Information Processing Systems 27*. 2014.

47. Han, Song, Huizi Mao, and William J. Dally. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.” In *International Conference on Learning Representations*. 2016. arXiv:1510.00149.

48. Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. “Distilling the Knowledge in a Neural Network.” arXiv:1503.02531, 2015.

49. Frankle, Jonathan, and Michael Carbin. “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks.” In *International Conference on Learning Representations*. 2019.

50. Kornblith, Simon, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. “Similarity of Neural Network Representations Revisited.” In *Proceedings of the 36th International Conference on Machine Learning*, 3519–3529. Proceedings of Machine Learning Research 97, 2019.

51. Olah, Chris, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. “Zoom In: An Introduction to Circuits.” *Distill*, 2020. DOI: 10.23915/distill.00024.001.

52. Elhage, Nelson, et al. “Toy Models of Superposition.” *Transformer Circuits Thread*, 2022. arXiv:2209.10652.

53. Meng, Kevin, David Bau, Alex Andonian, and Yonatan Belinkov. “Locating and Editing Factual Associations in GPT.” In *Advances in Neural Information Processing Systems 35*, 17359–17372. 2022.

54. Geiger, Atticus, Hanson Lu, Thomas Icard, and Christopher Potts. “Causal Abstractions of Neural Networks.” In *Advances in Neural Information Processing Systems 34*. 2021.

55. Geiger, Atticus, et al. “Causal Abstraction: A Theoretical Foundation for Mechanistic Interpretability.” *Journal of Machine Learning Research* 26 (2025). arXiv:2301.04709.

56. Nanda, Neel, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. “Progress Measures for Grokking via Mechanistic Interpretability.” In *International Conference on Learning Representations*. 2023.

57. Bricken, Trenton, et al. “Towards Monosemanticity: Decomposing Language Models with Dictionary Learning.” *Transformer Circuits Thread*, 2023.

58. Cunningham, Hoagy, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. “Sparse Autoencoders Find Highly Interpretable Features in Language Models.” In *International Conference on Learning Representations*. 2024. arXiv:2309.08600.

59. Gao, Leo, et al. “Scaling and Evaluating Sparse Autoencoders.” arXiv:2406.04093, 2024.

60. Ferns, Norm, Prakash Panangaden, and Doina Precup. “Bisimulation Metrics for Continuous Markov Decision Processes.” *SIAM Journal on Computing* 40, no. 6 (2011): 1662–1714. DOI: 10.1137/10080484X.

61. Zhang, Amy, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. “Learning Invariant Representations for Reinforcement Learning Without Reconstruction.” In *International Conference on Learning Representations*. 2021. arXiv:2006.10742.

62. Geiger, Atticus, Zhengxuan Wu, Thomas Icard, Christopher Potts, and Noah D. Goodman. “Inducing Causal Structure for Interpretable Neural Networks.” In *Proceedings of the 39th International Conference on Machine Learning*. Proceedings of Machine Learning Research 162, 2022.

63. Turner, Alexander Matt, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. “Steering Language Models with Activation Engineering.” arXiv:2308.10248, 2023.

The author has published over 5,700 AI-assisted papers at: https://www.researchgate.net/profile/Douglas-Youvan/research . Google Scholar has indexed these preprints, and if you want a particular reference, the AI mode of a Google search (Gemini) has efficiently catalogued these preprints.